DOCUMENT RESUME

ED 093 340 IR 000 827

AUTHOR	Nevison, John M.; And Others
TITLE	Moving the Program: The Final Report of Project CONDUIT at Dartmouth College.
INSTITUTION	Dartmouth Coll., Hanover, N.H. Kiewit Computation Center.
SPONS AGENCY	National Science Foundation, Washington, D.C.
PUB DATE	May 74
NOTE	168p.
EDRS PRICE	MF-$0.75 HC-$7.80 PLUS POSTAGE
DESCRIPTORS	*Computer Programs; Computers; Computer Science; *Demonstration Programs; *Higher Education; *Interinstitutional Cooperation; Program Descriptions; Programers; *Programing; Programing Languages; Research Projects; Time Sharing
IDENTIFIERS	Compatibility; Dartmouth College; National Science Foundation; *Project CONDUIT

ABSTRACT

A two-year experiment in educational computer usage and program exchange sponsored by the National Science Foundation is described in the body of this review. The purpose of the project was to set up a prototype organization of five university computer centers in order to establish a set of procedures that would facilitate the efficient sharing of computer programs among the five centers. The main document in the report briefly describes: the reasons for having this kind of exchange program; the mechanics of the organization that was set up; the progress made in the first two years of the project; a review of the problems encountered in the moving of programs; and a set of conclusions and recommendations for those who may be involved in the moving of programs from one computer to another. The fifteen appendixes include discussions of a variety of related problem areas, and, together with the introduction, provide a valuable text that should be useful to anyone working with computers in the academic community. (WDR)

CONDUIT

BEST COPY AVAILABLE

BEST COPY AVAILABLE

A consortium of regional networks at
Oregon State University, North Carolina Educational Computing Service,
Dartmouth College, and the Universities of Iowa and Texas (Austin).

Moving the Program: The Final Report of Project CONDUIT at Dartmouth College

John M. Nevison

Appendices of Collected Papers

Christopher G. Hoogendyk & John M. Nevison

May 1974

U S DEPARTMENT OF HEALTH
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

(NSF Grant Nos. GJ-31754/EC-31754
Principal Investigator: Thomas E. Kurtz

Errata for Moving the Program:The Final Report of Project Conduit at Dartmouth College

rors are referenced by page and line, in the form PP.LL

PORT

enters is" --> "centers"
ange period to comma after "cards"
ecline" --> "declined"
lete "completely"
xerpts" --> "Excerpts"
hows" --> "show"
osts" --> "cost"
einer" --> "Reiners"
eward" --> "rewards"
omputrized" --> "Computerized"
une" --> "May"
roject" --> "Project"
aylored" --> "tailored"
ayloring" --> "tailoring"

B: Marketing Mailouts

ge incorrectly numbered "B3"
.60" --> "3.75"

C: Mag Tape Format for Export

3 "standard" --> "guideline"
derline "only by"
40" --> "40"
lete comma after "roughly"
ndustry wide" --> "industry-wide"
sert "In all the examples, items shown in lowercase should
be replaced by appropriate names or codes for specific
cases." after first paragraph
pend comma to end of card "// VOL=SER=tpnam"
csl11" --> "discname"
et into" --> "deal with"
rote" --> "writes"
ould" --> "can"
lete period after "gap", insert period before "An"

D: Preparing a Program for Export

d comma after "documented"
primary ... readability" --> "A program should be readable."
s quickly" --> "quickly"
lete "as possible", delete "consideration"
oley" --> "solely"
replace with
1. Use a standard language
2. Program in modules
3. Structure your program
4. Use variables for recurring constants
5. Make programs easy to check
6. Comment your program clearly

sing" --> "A program written in"
ncreases the" --> "has a wider"
lete "of a program" delete "which is"
ther" --> "another"

2.37	"Programs" --> "A program" "modular form" --> "modules"
3.3	"modularly" --> "in modules"
3.26	"increases the readability of a program" --> "makes a program more readable"
3.36	"Parameterization should be used to eliminate" --> "Eliminate"
4.8	delete "the problems of"
4.13	"necessary documentation" --> "documentation necessary"
4.18	"word size incompatibilities" --> "differences in word size"
4.21	"a necessity" --> "is necessary"
4.27	delete "be used to"
4.35	delete "and be able to"
6.18	"with consideration for" --> "according to"

Appendix F: Agreeable Fortran

1.28	"agains" --> "against"
2.32	"parenthesis" --> "parentheses"
3.12	"an error message printed" --> "appropriate action taken"
3.12-14	"The action ... moved." --> "Various implementations will either fall through or abort when the variable is out of range. The results cannot be depended on between one system and another"
3.19	delete "large"
4.4	insert "It also makes a difference in transmission or filling of arrays using READ or DATA statements in which an array name implies all the elements of the array. The array may be filled by row or by column depending on the internal layout of the array. In such cases implied DO loops should be used with the array name."
4.24-25	move "of those which are used" to after "list"

Appendix G: Editorial Aids

2.2	delete comma after "RENAMEd"
3.3	"fater" --> "after"
3.25	"more" --> "more than one program unit (i.e. SUBROUTINEs or FUNCTIONs in the file, the format"
3.28	"use FORMAT" --> "use FORMATS"
4.1	change both periods to slashes
4.6	"Filename 2" --> "Filename2"
4.32	"file" --> "files"
6.3	"FORTRAN" --> "Fortran", "BASIC" --> "Basic"
7.4	"FORTRAN" --> "Fortran"
7.5	"BASIC" --> "Basic"
7.13	"cuase" --> "cause"
7.14	"FORTRAN" --> "Fortran", "BASIC" --> "Basic"
8.12	"FORTRAN" --> "Fortran" "BASIC" --> "Basic"
9.9	"FORTRAN" --> "Fortran", "BASIC" --> "Basic"
9.30	"After correction" --> "After correcting"

Appendix H: Programmer's Weapons

1.14	delete "of"

Appendix I: Knights in Rusting Armour

9.13	"tayloring" --> "tailoring"
14.3	"Reiner's" --> "Reiners"
15.4	insert "committee" after "discipline"
15.29	"This" --> "Such a"
12.24	"excerises" --> "exercises"

Table of Contents

Introduction 1

Lay of the Land 2

Two Fast Years: 1972-73 - 1973-74 3

The Soluble Problem: Moving a Program 5

The Insoluble Problem: Moving the Idea 8

Results 9

Conclusions 14

References 16

Appendix A 18
Technical Costs of Program Set Preparation (JMN)

Appendix B 22
Marketing Mailouts

Appendix C 27
Magnetic Tape Format for Exporting Programs (CGH)

Appendix D 46
Preparing a Program for Export (CGH)

Appendix E 56
Describing a Program for Export (JMN)

Appendix F 66
Agreeable Fortran (CGH)

Appendix G 73
Editorial Aids for Importing and Exporting Fortran Programs on DTSS (CGH)

Appendix H 85
The Export System (JMN)

Appendix I 89
How COMBIB Works (JMN)

Appendix J 97
COMPUTe Publications

Appendix K 123
CONLIB***:LIBCAT (D. Cooley, CGH)

Appendix L 131
CONDUIT Staff 1973-74

Appendix M 133
Programmer's Weapons (JMN)

Appendix N 138
Knights in Rusting Armour (Report without appendices) (JMN)

Appendix O 156
Dartmouth-CONDUIT
June 1, 1974 - December 31, 1974

CONDUIT

Kiewit Computation Center
Dartmouth College
Hanover, New Hampshire 03755

POLICY BOARD

LARRY C. HUNTER
Oregon State University

THOMAS E. KURTZ
Dartmouth College

LOUIS T. PARKER, JR.
North Carolina Educational Computing Service

CHARLES H. WARLICK
University of Texas

GERARD P. WEEG
Chairman
The University of Iowa

DIRECTOR

JAMES W. JOHNSON
CONDUIT/Central

Introduction: The Scope of the Project

The letterhead says CONDUIT. The places mentioned are Oregon State University,O; North Carolina Educational Computing Service, N; Dartmouth College, D; University of Iowa, UI; The University of Texas, T. Their common concern: Computers, C.

A line across the bottom of the page explains: "An experiment in educational computer usage and program exchange sponsored by the National Science Foundation." Experiment means that the project's organization of people is a prototype. The first interest of the Foundation is to discover if this prototype is an effective organization. The first interest of the computer centers is in this prototype is the exchange of programs.

These conflicting interests have been with the project since its beginning in January of 1972 when one person at each center was designated Coordinator. His (or her) tasks were bipartate: first, move programs; secondly, report on their movement.

A project director with a staff of three was established to ride herd on the dispersed coordinators. A psychological testing agency prepared to monitor the project's activity. The prototype was set: a cooperative of representatives from dispersed computer centers coordinated by a small central office.

To provide an immediate focus for moving programs, six workshops were organized. To each workshop came two professors from each of the five regions. After the workshops the teachers returned home to try out the new ideas with their students. The computer programs accompanying these ideas were to be implemented in each region by the Coordinator and his staff. During the summer and fall of 1972, educational uses of the computer in Linear Algebra, Topics in Physics, Numerical Methods in Chemistry, Operations Research, Economic Games, and Survey Analysis were each the subject of a workshop.

By June of 1973, students and teachers had used and endorsed these several and various methods of using computing in their courses (1). The project staff, however, recognized that serious differences between the five centers were inhibiting the flow of programs.

Differences over the mailing of magnetic tape and punched cards. Fortran and Basic dialects, standards for program documentation, and protocols for sending and receiving programs, all posed problems that could be solved.

The second year of the project saw an organized effort to solve these technical difficulties. By the end of May, 1974, several documents published by the Central Office attempted to codify this effort (2-6).

In addition to the import and export of programs, each center made a collateral effort to find good new applications, to publicize them, and to encourage their use by whatever means seemed appropriate. This broad charge was freely interpreted by different individuals in different ways. One common thread running through the activities of all five computer centers was the regional workshop, a one-day to one-week conference to promote a new idea in computing.

While the project was funded from January, 1972 through December, 1974, its activities focused on the academic years 1972-73 and 1973-74. This report is the final report of the Dartmouth office of Project CONDUIT and covers the period from the beginning of the project through May 31, 1974. The last appendix will contain late comments on the period June 1, 1974 to December 31, 1974.

The Lay of the Land

Before discussing the activities of the Dartmouth CONDUIT office, it is necessary to explore the terrain in which these activities were to take place.

The computing environment is conditioned by three fundamental convictions:

1) Computing should be freely available to the whole campus community;
2) Computing should be fast and easy to use;
3) Remote users (customers) should receive the same service from the machinery that on-campus users receive.

These convictions led to:

1) The development of a language, BASIC, that can be learned in a few hours; and

2) The development of a time-sharing system with several hundred terminals available at the user's convenience.

When an individual works at a terminal, he can expect the computer to respond to his commands within less than four seconds. The computer is extremely reliable; it operates smoothly more than 99% of its scheduled 8:00 a.m. to 3:00 a.m. day.

Because the computer is reliable and virtually everyone interested in computing on campus is capable of getting on the computer, almost all services to the user may be obtained on-line. There is a library of programs with files explaining their use. There are programs that explain many aspects of the system. There is an on-line suggestion box where answers are guaranteed within twenty-four hours.

The result of the BASIC language, the time-sharing system and the on-line services is a campus population literate in computing. Some of the faculty and the majority of the students know how to write simple programs. A few remote colleges also have significant groups of students and faculty who are literate in computing.

Two Fast Years: 1972-73 - 1973-74

Not surprisingly, a few teachers have explored ways to use this new activity, computing, in their research and in their teaching. John Merrill in the Physics Department at Dartmouth took advantage of support from Project COEXIST to write a text entitled The Computer in Second Semester Physics. This text touches a variety of topics that crop up in Physics from the second semester on through graduate work: mechanics, magnetic fields, relativity, to name a few.

In June of 1972, John Merrill taught a two-week Physics Workshop. (This workshop is often called in the CONDUIT publications a "national workshop" to distinguish it from a regional conference, "local workshop," held by a participating computer center.) Upon its conclusion, teachers returned to Iowa, Texas, Oregon and North Carolina to begin the year's work with his Physics programs. A complete report of this workshop is on file in the Central Office (7). An edited version has been published by that office (8).

By September, the Dartmouth office was importing programs to help the small band of eight teachers that had attended the National Workshops. This was not the full complement of twelve. Missing was a second mathematician, a second economist, and two teachers of operations management. Of the eight teachers, four - contrary to the original plan - were not being served by the Dartmouth computer. Robert D. Gillette taught Chemistry at the Naval Academy. All ties to him were through the U.S. mails. Similarly, Barrien Moore in the Mathematics Department, and John Dawson and Richard St. Onge in the Physics Department of the University of New Hampshire were all cut off when the university severed its ties to Dartmouth and began working on their own computer in the fall of 1972.

The four remaining teachers were served by the Dartmouth computer:

James Boyles - Chemistry Department, Bates College
Elinor Hartshorn - Political Science Department, Mount Holyoke College
George Michaelides - Economics Department, Franklin Pierce College
David Rosenberg - Political Science Department, Middlebury College

The trials and tribulations of the first year have been fully reported in the Interim Report, Knights in Rusting Armour (9). (The body of this report is Appendix N.)

During the second academic year of the project, import activities decline, export activities increased, and marketing efforts slumbered. With no new round of workshops to stimulate demand for imported programs, the volume of imported programs dwindled. Two large programs were successfully begun: Prince, an international politics game from Syracuse, and a series of simulations in Epidemiology from Cornell (See (10-14) (17) and Appendix A). Each was undertaken at the request of a professor at Dartmouth. Two more data sets were prepared for use on the IMPRESS system, and two more workbooks were prepared for use. (See Appendix A.)

Some of the first year's work was consolidated in drafts of future booklets. James Boyles at Bates prepared some notes for his students using MASSPEC, "An Experiment in Mass Spectrometry Utilizing a Computer Simulation of Mass Spectrometer." David Rosenberg completely wrote a workbook for his students using the IMPRESS dataset, CITIZENS. The booklet was entitled "Citizens and the Political Attitudes in Four Countries."

The number of exported programs increased because of the need for trial cases for technical standards. Three program sets were prepared to demonstrate the on-line export system (discussed below). CONDUIT also began to revise programs for COMPUTe texts.

The marketing activities slumbered because of a four-month absence of the project Coordinator. The available program sets were announced in the October issue of the center's newsletter. They were announced again a month later in a letter to the faculty. (See Appendix B.)

A quite successful Regional Conference on Computing took place in January. Some tape slide shows were made to demonstrate new uses of computing. An on-line bibliography of texts and booklets was polished and put into smooth operation. (See (18,19) Appendix I.) Contacts between the commercial publishers and the local office increased and the groundwork was laid for future cooperation (15). In May an advertisement of the Huggins text was sent to over 1300 people. (See Appendix B.)

The solution to the technical problems began when the Chief Programmer from Dartmouth visited all five CONDUIT centers in July of 1973. His experience with magnetic tape and Fortran formed the basis for the first Programmer's Meeting. The meeting was held at Dartmouth on October 3-4, 1973. Subsequent meetings of programmers and coordinators sped the development of technical agreements. At Dartmouth, the coordinator and the chief programmer collaborated on a series of local working papers describing various aspects of this work.

The second year's activity very clearly separated the CONDUIT problems of transporting educational programs into two kinds. There were problems CONDUIT could work through and the problems CONDUIT had to work around.

The Soluble Problem: Moving a Program

> "Moving a program from one computer to another is like moving a horse by train: fundamentally it does not want to go." (16)

The first year of the project illustrated, if nothing else, how not to transport computer programs. The first Interim Report treated in detail the case of some badly moved Chemistry programs. (See Appendix N, pp. 7-9.) The following mistakes occurred in one or more of those programs:

..programs that produced incorrect answers were selected;

..programs that did not run were exported;

..character codes were not shipped with magnetic tapes of programs;

..printed copy of the programs was not shipped with the magnetic tape;

..flow charts and sample data and sample runs were not sent with programs.

The first two mistakes wasted forty hours of a very skilled programmer's time. The next three mistakes cost another forty-two hours of wasted effort.

The second year solved these problems completely. Using a standard character code, and a standard magnetic tape format, (See Appendix C), a programmer successfully loaded four program sets from the four CONDUIT centers. Each set was loaded in fifteen minutes. After a set of programs was loaded, each individual program imported to Dartmouth was translated into BASIC. Exerpts from the first year's report shows what some of this effort costs (9, p. 3):

	Computing Cost	Programmer Cost ($2.00/hr)	Total
Business	$342.18	$169.50	$511.68
Linear Algebra	60.06	57.00	117.06

Yet these efforts, while satisfying those who had attended workshops and were familiar with the programs, were not complete. To satisfactorily document these programs (internally) would have required additional hours of programmer time.

For example, both the Business and the Linear Algebra programs were embedded in a text. The Linear Algebra programs did not contain (and did not need, if used with the text) any internal documentation. However, if these programs were to be placed in a public library where they could possibly be used independently of the text, they would require the following extra work.

	Computing Cost	Estimated* Extra work	New Total
Business	$511.68	$20.00	$531.68
Linear Algebra	117.06	24.00	141.06

*(at $2.00/hr)

These costs could be avoided if programs were properly prepared for export. So, the programmers from the five centers established working agreements on two programming languages, FORTRAN and BASIC, (See (5) and Appendix F.) and on ideal programming style, internal program documentation, and the accompanying commentary. (The Dartmouth version appears in Appendicies D and E.)

The Dartmouth office applied these standards to three program sets it prepared for export. The resulting costs are listed in Table 1.

Table 1

Program Sets For Export
(Figures for May 15, 1974)

	Huggins	Merrill	Reiner
No. of Programs	2	60	16
Program	1	59	8
Q-File	1	1	8
Lines of Program	468	5333	1600
Code	71	1762	331
Internal Comments	25	1684	403
Q-File	372	1887	866
Preparation Costs	$192.92	$316.53	$477.88
Programmer ($3.00/hr)	$ 84.00	$165.00	$231.00
Computer	108.92	151.53	246.88
Languages	BASIC	BASIC,FORTRAN	BASIC
Price/Set	$ 15.00	$ 25.00	$ 35.00

The costs of the Merrill set is significantly less than the Reiner set because it was prepared after the standards for programs and comment files had been distributed to the local programmers.

Another soluble problem was the routine export of programs. During the first year all requests had been handled on an *ad hoc* basis. During the second year of the project a fully computerized export system was implemented. It first ran successfully on April 15, 1974. (See Appendix H.)

To help in the Fortran-to-Basic translation of the first year, a set of editorial aids were written. (See Appendix G.) During the second year some additional aids for translation from Basic to Fortran were developed.

The Insoluble Problem: Moving the Idea

More important than moving a program is moving the idea of the program. The idea of the program is what the teacher or researcher values. He must have a clear explanation of that idea before he is even willing to look at the program, let alone think of obtaining a working version.

If an idea is going to travel any further than its originator, it must be written down. When the idea involves a computer program used in college instruction, what gets written down generally takes the form of a text, or text supplement. To write such a booklet will, in the current scheme of tenure calculations, fail to advance an author's career and may retard it. So the first snag is

the rewards for the author.

Dartmouth CONDUIT worked around this problem by encouraging the efforts of a sister project, COMPUTe. COMPUTe provides authors with royalties for their efforts and produces much needed texts and text supplements on instructional computing. (See Appendix J.) CONDUIT will be responsible for establishing standards for the computer programs in new COMPUTe texts and has undertaken to revise some of COMPUTe's current offerings.

The local CONDUIT office has also shown manuscripts to commercial publishing firms who could move texts by traditional methods. To date these efforts have been unsuccessful. (For reasons, see (15).) After a book is published, it needs to be marketed. CONDUIT, through its five center connections, can contribute to this marketing of a booklet.

But as one approaches the marketing of a booklet, one sees several other insoluble problems. What one markets is the idea of the program. For a simple notion, a short program can be moved in a text. For a more complex idea, for example, a game the size of PRINCE, the program itself is the expression of the idea, and it must be moved. (The real achievement of CONDUIT has been to solve most of the problems associated with this movement.) But either the simple or the complex idea needs to be adjudged a good idea before it will be used. This judgement depends heavily on

the convenience of computing.

The convenience of computing is a function of the local computer and the services of the computer center staff. The more inconvenient the computing, the greater the hardships for a computing idea as it competes for the instructor's attention. Even with convenient computing, an idea must fit in with

the content of the course.

How a teacher composes his course determines what he decides he will need to teach it. Here, a computing idea must compete like all others for its proper audience. After Project CONDUIT removes any handicaps by making sure the associated programs work, the idea must then sell itself. And in selling itself, two more insurmountable problems confront it:

the ability of the teacher and
the ability of the student.

There will always be bright students and dull students. Nothing can be done about that. But what computation centers can do, again, is remove the computing idea's handicaps by training students and teachers in computing. Were this an accomplished fact, then "ability" would mean academic ability not computing ability. Project CONDUIT, by encouraging the computer center efforts to raise the computing literacy on campus, aids the struggling computing idea.

Results

The four major results of the Dartmouth CONDUIT office are:

Moved Computer Programs
Reports on Activities
Working Papers
Computrized Systems

Moved Computer Programs

Programs fall into categories that deserve some explanation. First are programs available in the on-line library, CONLIB.(See Appendix K.) In this library are 73 programs made available by Project CONDUIT to users of the Dartmouth Time-Sharing System. Most of these programs were not only imported, but translated from Fortran to Basic. Many of these programs form sets for a text. The following texts have 41 programs in CONLIB:

(10/10 programs	Harris, Roy D. and Michael J. Maggard, Computer Models in Operations Management, Harper and Row, New York, 1972.
(18/21 programs)	Johnson, K. Jeffrey, Numerical Methods in Chemistry, (2nd Edition) Chemistry Department, University of Pittsburgh, 1973.
(5/5 programs)	McLaughlin, Donald E., A Computer Oriented Course in Linear Algebra, Regional Computer Center, University of Iowa
(8/8 programs)	Reiners, William A., William E. Glanz, and Stanley E. Cornish, Ecological Modeling on the Dartmouth Time-Sharing System, (3rd Edition), Biology Department, Dartmouth College, 1973.

This leaves 32 additional loose programs in economics and chemistry.

Not listed in CONLIB but translated by Project CONDUIT are the five data bases for four social science workbooks used on the IMPRESS system:

1972-73

(1 data base)	Changing Attitudes Toward Integration 1946-1966 (IMPRESS Edition), G.R. Boynton, University of Iowa, 1972.
(2 data bases)	Voting Behavior in Western Europe, (IMPRESS Version), Gerhard Lowenberg, University of Iowa, 1973.

1973-74

(1 data base)	Voting Behavior in the United States: 1952-1968, (IMPRESS Version), G. R. Boynton, 1972.
(1 data base)	Citizens and the Political System (IMPRESS Supplement), to Citizens and the Political System, G. R. Boynton, Harper & Row, New York

The Dartmouth office promoted the casual export of the following books:

1. Bauer, Richard S., *Introduction to Time-Sharing Applications in Banking*, Kiewit Computation Center, Dartmouth College, 1974.
2. Huggins, Elisha, *A Computer-Assisted Introduction To Mechanics*, Kiewit Computation Center, Dartmouth College, 1973.
3. Merrill, John R., *The Computer in Second Semester Physics*, Kiewit Computation Center, Dartmouth College, 1972. (To be published by Houghton Mifflin.)
4. Olson, Maynard, *Computer Calculation of Chemical Equilibrium*, Chemistry Department, Dartmouth College, 1973.
5. Reiners, William A., William E. Glanz, and Stanley F. Cornish, *Ecological Modeling on the Dartmouth Time-Sharing System*, (3rd Edition), Biology Department, Dartmouth College, 1973.
6. Sokol, Robert, *A Laboratory Manual for Introductory Sociology*, Harper & Row, New York.
7. Williamson, J. Peter and David H. Downes, *Manual for Computer Programs in Finance and Investments*, (3rd Edition), Amos Tuck School of Business, Dartmouth College, 1973.

Computer programs for Olson are contained in the text. Those for Williamson and Bauer are distributed by the business school and the computation center. Sokol's programs work on IMPRESS and on data cards. The three other authors served as test cases for the new documentation standard and export system. These three program sets are now for sale.

$15.00	Huggins	(2 programs)
$25.00	Merrill	(60 programs)
$35.00	Reiners	(8 programs)

Reports of Activity

The project has produced nine reports listed below. Together they detail many of the experiences of one office involved in a prototype cooperative.

Table 2

Reports of the Dartmouth-CONDUIT Office

1. A Report on the CONDUIT Physics Workshop Held June 19-30, 1972 at Dartmouth College, John M. Nevison. Internal Memorandum Submitted in August of 1972 to CONDUIT Central Office at Duke University.

2. Report on the December 9th Social Science Workshop at Dartmouth, John M. Nevison. Internal Memoranda submitted December 20, 1972 and March 26, 1973 to the CONDUIT Central Office at Duke University.

3. Report on a Workshop: The Kiewit-CONDUIT Conference March 16-17, 1973, John M. Nevison. Internal Memorandum submitted May 17, 1973 to the CONDUIT Central Office at Duke University.

4. Report on a Workshop: IMPRESS at Wellesley 21 April 1973, John M. Nevison. Internal Memorandum submitted May 18, 1973 to the CONDUIT Central Office at Duke University.

5. Leading the Horse to Water: A Report on the CONDUIT-HumRRO trial of a kit of social science booklets by users of the Dartmouth Time Sharing System, John M. Nevison. Internal Memorandum submitted in August 1973, to the CONDUIT Central Office at the University of Iowa.

6. Classroom Report: Political Science, IMPRESS, and CONDUIT at Middlebury 1972-1973, David A. Rosenberg, July 19, 1972, submitted in October 1973 to CONDUIT Central Office at the University of Iowa.

7. Knights in Rusting Armour: A Report on the CONDUIT Activities at Dartmouth, June 1972-May 1973, (Interim Report), John M. Nevison, October 1973.

8. Report on the Kiewit-CONDUIT-COMPUTe Conference. January 25-26, 1974. Christopher G. Hoogendyk. Internal Memorandum submitted to CONDUIT Central Office, Iowa City on June 7, 1974.

9

9. Project CONDUIT at Dartmouth College (Final Report), John M. Nevison, June 1974.

Working Papers

During the second year, standards for the office operation were documented before, during, and after the fact. The eight working papers listed below specify the ideal operation of an import-export office.

Table 3

Working Papers of the Dartmouth-CONDUIT Office

1. Preparing a Program for Export, Christopher G. Hoogendyk, April 1974.
2. Describing a Program for Export, John M. Nevison, April 1974.
3. Agreeable Fortran, Christopher G. Hoogendyk, May 1974.
4. Editorial Aids for Importing and Exporting Fortran Programs on DTSS, Christopher G. Hoogendyk, October 1973.
5. The Export System, John M. Nevison, May 1974.
6. How Combib Works, John M. Nevison, May 1974.
7. Magnetic Tape Format for Exporting Programs, Christopher G. Hoogendyk, May 1974.
8. Programmer's Weapons, John M. Nevison, May 1974.

Computerized Systems

Last and most important are the on-line systems that increased the office's efficiency. The major systems are discussed in detail in the working papers:

1. The on-line library and its table of contents, LIBCAT. (See Appendix K.)
2. The complete reference system for texts and programs on instructional computing, COMBIB. (See Appendix I.)

3. The set of programs that provides editorial aid in program translation, import, and export. (See Appendix G.)

4. The system that makes possible prompt and versatile program export. (See Appendix H.)

Conclusions

A number of programs shipped to the computer center were not moved to class because they lacked an adequate description of their use in the course. Of those that had adequate descriptions, only a few, were innovative enough to command any interest. Two lessons emerge from this:

1. A computer center should not import a program without an adequate description of how to use it in a course.

2. A computer center should not import a program unless a teacher has requested it.

A variety of technical problems made the original import of some programs difficult. Had the programs been properly prepared for shipment this would not have occurred. The third lesson of the Project is

3. A computer center should not import programs that are poorly prepared for shipment.

The other side of that lesson is

4. A computer center should not export programs improperly prepared for shipment.

"Proper" preparation has been defined in the working papers of the project. (See especially Appendices D and E.) Indeed, this definition is the major success of the project.

Work with these properly prepared programs has shown:

5. It is possible to set useful standards for exporting a computer program.

6. It is possible to export a program using these standards.

7. Such a prepared program is at least an order of magnitude faster and cheaper to import (15 minutes versus 42 hours).

Program sets like those associated with the Prince books and the Epidemiology simulations were imported without benefit of the local hosts having attended any workshops. This experience indicates that

8. An adequate written description of use in the course is more important to a program's dissemination than is a workshop. (It is also much cheaper.)

Last and most important, good ideas did move from the computer center into the classroom. There, each teacher had his own opinions about how to use a program and one observes that

9. On an interactive computer, a library program will frequently be taylored to the needs of a particular course.

While this tayloring was sometimes done by the teacher, one of his students often did it for him. This practice was so common that one can say that

10. The student-teacher pair is a most effective team for bringing a computing idea to its ultimate fruition in the course.

From one office's view point the Project succeeded. It moved programs and reported on their movement. It established local operating procedures that employed computerized systems when appropriate. It defined standards for program export and produced examples of these standards. In short, it solved those problems within its focus and illumined those on its periphery.

References

1. Culp, George H. and James Johnson. The CONDUIT Study, Transportation and Dissemination of Computer Related Curriculum Materials. Preliminary Findings, University of Texas Computer Center, Austin, Texas, February 1974. 6 pp.

2. CONDUIT Tape Distribution Form. CONDUIT Central, Iowa City, Iowa, November 1973. 2 pp.

3. Dunnagan, Trinka. CONDUIT Documentation Guidelines: Abstracted Version. CONDUIT Central, Iowa City, Iowa, October 1973. 2 pp.

4. Dunnagan, Trinka. CONDUIT Documentation Guidelines. CONDUIT Central, Iowa City, Iowa, October 1973.

5. Dunnagan, Trinka (ed.). CONDUIT Technical Transport Guidelines. CONDUIT Central, Iowa City, Iowa, May 1974.

6. Anderson, Ron and Trinka Dunnagan. CONDUIT Review Rating Form. CONDUIT Central, Iowa City, Iowa, November 1973.

7. Nevison, John M. A Report on the CONDUIT Physics Workshop Held June 19-30, 1972 at Dartmouth College. Internal Memorandum submitted August 1972 to CONDUIT Central at Duke University.

8. 1972 CONDUIT National Workshops, Combined Preliminary Reports. Internal Memorandum, CONDUIT Central, Iowa City, Iowa, May 1973.

9. Nevison, John M. Knights in Rusting Armour: A Report on the CONDUIT Activities at Dartmouth June 1972-May 1973. 27 October 1973.

10. Coplin, William D. and Michael K. O'Leary. Everyman's PRINCE: A Guide to Understanding Your Political Problems. (North Scituate:Duxbury Press, 1972) $2.95

11. Handelman, John, William D. Coplin, Stephen L. Mills, and Michael K. O'Leary. International Relations Theory Through PRINCE: A Workbook. (Test Edition) International Relation Program, Syracuse University, July 1973. $2.00

12. Coplin, William D., Stephen L. Mills and Michael K. O'Leary. PRINCE Participant's Guide: Concepts, Environment Procedures. International Relations Program, Syracuse University, March 1972. $1.25

13. Handelman, John R., William D. Coplin, Michael K. O'Leary and Bruce Riddle. Instructor's Guide to PRINCE Educational Materials. International Relations Program, Syracuse University.

14. Coplin, William D., John Handelman, Stephen L. Mills and Michael K. O'Leary. A Description of the PRINCE Model. International Relations Program, Syracuse University. $2.00

15. Nevison, John M. "To Know in Part: CONDUIT, Publishers and Computer Vendors." SIGCUE Bulletin, Association of Computing Machinery, March 1974, pp. 15-19.

16. Nevison, John M. "Horse Sense." Computer Decisions, (to be published)

17. Epidemic Simulation for Students in Medicine. Center for Environmental Quality Management, Cornell University, Ithaca, New York, (Reprint 1033) 1972.

18. Nevison, John M. (ed) "Project CONDUIT On-Line Bibliography." SIGCUE Bulletin, Association of Computing Machinery, October 1973, Vol. 7, No. 4, pp. 27-34.

19. Nevison, John M. (ed) "CONDUIT On-Line Bibliography." ON LINE, University of Michigan, May 1973, Vol. 3, No. 3, pp. 27-40.

APPENDIX A

Technical Costs of Program Set Preparation

The Problems of Prince

Prince is an international diplomacy game, written in Fortran. At the request of Professor David Baldwin, the Dartmouth CONDUIT office imported the program for his government course.

In April, 1973, the program was purchased from the International Relations Program at the University of Syracuse for $50.00. When it arrived there were problems reading the magnetic tape. Several phone calls to Syracuse cleared up these problems.

The second set of problems were caused by the lack of suitable documentation of the program. There was no flow chart, no internal program documentation, and no usable sample output. After a series of phone calls, the description of the program (a flow chart of sorts) arrived with some sample runs of the Syracuse version. All this engaged a programmer for about 40 hours.

His next 46 hours were spent doing battle with the program, the local problems with Dartmouth Fortran, and the core memory restrictions of time-sharing. The sub-program shifting in the Fortran run-time package would not handle more than 24 sub-programs. After making extra efforts to demonstrate that the error was in Fortran and not in the program, the programmer convinced the student in charge of Fortran to clean up his act. The programmer went home for the summer after 86 hours of work.

In October, Fortran ran better and sub-programs were well handled. The programmer set about the next major task: shoehorning a 30K (K is 1024 36 bit words of core memory) program into a 16K slot. First, he reduced the COMMON used by the program and shrank that work space from five thousand words to two hundred words. Next he broke up sub-programs into smaller units. Last, he closed each file as he was done using it and picked up 500 words for each shut file. The program finally ran in 16K of core.

What was saved in space was lost in run time. A 24K version ran in 50 seconds of CPU, but the 16K version ran in 200.

The October-November efforts of the programmer took 80 more hours. At this point, the program was demonstrated to David Baldwin and turned over to him for final polishing and use in his class. He hired a student to work on these final details. At least one student did use the program in his class this winter. More work is scheduled to be done this summer.

Programmer Costs	$498.00
Computer Costs	200.00

One Program

Lines of Program	1810
Code	1775
Comments	35

CONBIB

The actual programming of the on-line bibliography called CONBIB went quickly The coordinator and the programmer devoted a Saturday morning to the final design of the visible system. The programmer wrote the essential program in 15 hours. Editorial features were buit into the system in another 15.

A second programmer developed an "apple-picker" program to produce both a catalogue and an annotated bibliography from the file. His efforts eventually ran to 40 hours.

A third programmer prepared an alternate CONBIB to the director's specifications in five hours. This version was never examined for use by the director and was put aside.

The schedule of events was impressive. The program ran three days after the early June Saturday morning. The editorial features ran completely a week later. In July a general invitation to use the system was mailed to others in CONDUIT. The coordinator sent out a catalog generated by the "apple-picker" in late September. The Director's suggestions were made in early November and a working version was completed December 2nd.

The computer costs of this effort are crude estimates because the files that it used remained on-line and development efforts diffused through several user numbers.

Programmer Costs	$291.00
Computer Costs	370.00

Export

The *Export* system began in December of 1973. The coordinator specified the way it had to look to the user, and a programmer was charged with designing the system. (And altering the coordinator's original user specifications if they could be improved.) Over 50 hours went into designing and drafting a preliminary version of the program. It was then rewritten, entered and debugged in another 23 hours.

Programmer Costs	$219.00
Computer Costs	100.00

Program Sets for Sale

After establishing suitable standards for an exported program and its accompanying Q-file of comments, the staff applied these guides to three sets of programs. Two, Reiners and Huggins, were already complete by April 10th. Both needed minor additional revisions to meet the standards.

Merrill's set was begun and completed after April 10th. The programmer had the advantage of knowing exactly how the programs should be prepared *before* he began to work.

Program Sets For Export
(Figures for May 15, 1974)

	Huggins	Merrill	Reiner
No./Programs	2	60	16
Program	1	59	8
Q-File	1	1	8
Lines/Program	468	5333	1600
Code	71	1762	331
Internal Comments	25	1684	403
Q-File	372	1887	866
Preparation Costs	$192.92	$316.53	$477.88
Programmer ($3.00/hr)	$ 84.00	$165.00	$231.00
Computer	108.92	151.53	246.88
Languages	BASIC	BASIC,FORTRAN	BASIC
Price/Set	$15.00	$ 25.00	$ 35.00

Appendix A - Table

Program Set Preparation

Technical Costs 1 June 1973-31 May 1974

	Computer Cost	Prog. Costs ($3.00/hr)	Total Technical Costs
Import			
PRINCE	$200.00	$498.00	$698.00
SOCIAL SCIENCE	Not Avail.	462.00	462.00
In-House			
CONBIB	370.00	291.00	661.00
EXPORT-SYSTEM	100.00	219.00	319.00
Export			
HUGGINS	108.92	84.00	192.92
MERRILL	151.53	165.00	316.53
REINERS	246.88	231.00	477.88
TOTALS	1177.33 (+NA)	1950.00	3127.33 (+NA)

APPENDIX B

Marketing Mailouts

CONDUIT

KIEWIT COMPUTATION CENTER
DARTMOUTH COLLEGE
HANOVER, NEW HAMPSHIRE 03755

LICY BOARD

RRY C. HUNTER
gon State University

OMAS E. KURTZ
tmouth College

UIS T. PARKER, JR.
rth Carolina Educational
mputing Service

ARLES H. WARLICK
versity of Texas

RARD P. WEEG
versity of Iowa

November 8, 1973

Dear Colleague,

On the reverse side of this sheet is reprinted an article from the October issue of Kiewit Comments, the Computer Center's monthly newsletter.

I would like to take this opportunity to call the article and Project CONDUIT to your attention and explain what services , we can provide.

The Project itself is a prototype cooperative of five computer centers located at the schools of the Policy Board members on the letterhead. The cooperative is charged by the National Science Foundation with exchanging educational computer programs and learning from this exchange how to improve transport of programs. Needless-to-say the intent of this program exchange is to improve the quality of instruction.

The programs and the texts moved last year are all listed on the computer: Programs in CONLIB***:LIBCAT (LIST) and texts in CONLIB***:CONBIB (RUN). (Please feel free to suggest additional texts.)

Presently the Project has a staff of eleven part-time student (mostly) programmers. These programmers, headed by Christopher Hoogendyk, might possibly be of some help to you. Should you have any computer programs to import or export and want assistance please give me a call at 646-2643. (Best time is 1:00-3:00 pm.)

In addition to the programming staff, the Project is preparing several guides for those who import and export programs. A tape standard and character code has already been established for the cooperative and has proved very successful. Guides on programmer protocols and programming languages are in preparation.

I would be happy to answer any questions you might have on the work of the Project. In the event that I am out of town, Christopher Hoogendyk or Steve Waite, the Coordinator of User Services, should be able to help.

Sincerely,
John M. Nevison
John M. Nevison
Administrator, Project CONDUIT

An experiment in educational computer usage and program exchange sponsored by the National Science Foundation

Kiewit Comments *Vol. 7 No. 2* *October 1973*

CONDUIT AND THE DARTMOUTH SYSTEM

Users of the Dartmouth Time-Sharing System wishing to import computer programs for use in courses can use the facilities of CONDUIT, an experiment in exchanging educational computer programs. The project's coordinator at Dartmouth is Jack Nevison, who can be reached through Kiewit (603-646-2643), and there are available through the project a number of student assistants who have had experience in importing programs from other computing systems. The programs resulting from their efforts are described in CONLIB***:LIBCAT, which can be **LIST**ed. Other foci of CONDUIT are in Iowa, North Carolina, Oregon, and Texas, where there are IBM and CDC machines. CONDUIT has been involved in exporting programs from Dartmouth to these centers as well as importing material from them.

During this school year, CONDUIT has some funds to assist in paying student programmers who can work with faculty members in importing programs from outside Dartmouth. Those programs and texts which have been approved by CONDUIT committees in various disciplines will receive preference. A bibliography of books pertinent to the project can be obtained by **RUN**ning the file CONLIB***:CONBIB. Below is a partial list of the approved texts, arranged by discipline; more information about them and the names of people on the Dartmouth system who could answer some questions about them can be obtained from Nevison at Kiewit.

Business: *Computer Augmented Cases in Operation and Logistics Management*, by William L. Berry and D. Clay Whybark (Cincinnati; Southwestern Publishing Company; 1972); some of the cases are running in BASIC. *Computer Models in Operations Management*, by Roy D. Harris and Michael J. Maggard (New York; Harper and Row; 1972); all of the models are running in BASIC. Those interested in these programs can contact Steven Lazarus at the Tuck School (603-643-3292).

Chemistry: *Numerical Methods in Chemistry* by K. Jeffrey Johnson (Pittsburgh; Chemistry Department, University of Pittsburgh; n.d.); all the FORTRAN programs in the book are now available on DTSS in BASIC. *Computer Calculations of Chemical Equilibria*, by Maynard Olson (Hanover; Department of Chemistry, Dartmouth College); includes easy-to-write BASIC programs. Interested chemists can contact James Boyles at 207-784-4141 for information about the use of Johnson's text and Maynard Olson at Dartmouth (603-646-2763) for information about his own text.

Mathematics: *A Computer-Oriented Course in Linear Algebra*, by Don McLaughlin (Iowa City; Regional Computer Center, University of Iowa; n.d.); all the FORTRAN programs in this book are available in BASIC. Information is available from Barien Moore at the University of New Hampshire (603-862-2321).

Physics: *The Computer in Second Semester Physics*, by John Merrill (Hanover; Kiewit Computation Center, Dartmouth College; not yet published); has simple BASIC programs. *A Course in Computer-Assisted Mechanics*, by Elisha Huggins (Hanover; Kiewit Computation Center, Dartmouth College; n.d.); suggests simple BASIC programs. Merrill's work is being used at the University of New Hampshire by John Dawson (603-862-1395) and Richard St. Onge (603-862-2088) and at Dartmouth by Elisha Huggins (603-646-2969). Huggins can also be contacted for information about his text.

Social Science: *A Laboratory Manual for Introductory Sociology*, by Robert Sokol (New York; Harper and Row; n.d.); all the exercises in the book run on the IMPRESS*** system, and information about their use can be obtained from Sokol at 603-646-2637. Additionally, several workbooks describing elementary survey analysis have been translated from the University of Iowa's Statistical Package for the Social Sciences into Dartmouth's IMPRESS***; booklets published by the Regional Computer Center, University of Iowa, Iowa City, and available through the Dartmouth Bookstore are:

Voting Behavior in Western Europe, by Gerhard Lowenberg; and *Changing Attitudes Towards Integration: 1946-1966*, *The Citizen and the Political System*, and *Voting Behavior in the United States: 1952-1968*, all three by G. Robert Boynton. Some of these booklets have been used by David Rosenberg at Middlebury College (802-388-2901); they are also described in the library file CODEBOOK***.

In economics, some programs have been translated into BASIC, but they need verification by an economist working with with a student programmer. Nevison can supply details at Kiewit.

May, 1974

Dartmouth College HANOVER · NEW HAMPSHIRE · 03755

Kiewit Computation Center

ANNOUNCEMENT

Computer Assisted Introduction to Mechanics

(270 pp. - $3.~~60~~)
75

Elisha R. Huggins, Dartmouth Project COEXIST

This text, available in paperback from the Kiewit Computation Center at Dartmouth College, is the result of the first semester mechanics effort of Project COEXIST. Through a coordinated use of text, laboratory, and the computer, the book introduces Newtonian mechanics in six chapters. It can serve as a short text on mechanics or be used as a text supplement in a regular physics course. Since the main benefit of a computer in physics is that a student can work a realistic problem of his own choice, we have a special chapter on student project work. Special plotting techniques allow all the programs to run on a minicomputer as well as on a time-sharing system.

Table of Contents

Chapter I	Introduction to BASIC
Chapter II	Graphical Analysis of Strobe Photographs
Chapter III	Computer Analysis of Projectile Motion Experiments
Chapter IV	Newtonian Mechanics and Earth Satellites
Chapter V	Project Work
Chapter VI	Special Topics

The first chapter is a brief introduction to the computer language BASIC. This chapter is written so students can learn both the computer language and the plotting techniques on their own, without the assistance of an instructor.

The text approaches mechanics through a coordinated use of strobe photographs and the computer. In both cases, the motion of an object is broken down into short but finite time steps. We first use strobe photographs to help the student visualize this step by step description of motion, and then introduce a computer algorithm which allows the student to convert his strobe analysis into powerful techniques for predicting motion. Since we remain with short but finite time steps, calculus is not required for the student using this text.

Chapter II introduces the concepts of velocity and acceleration through a graphical analysis of strobe photographs of projectile motion. The chapter describes the laboratory procedures so that one can have students either take their own strobe photographs or use samples supplied in the text.

In Chapter III the graphical definitions of velocity and acceleration are converted to the basic steps or algorithm for a computer prediction of the motion of projectiles. Programs aid the study of projectile motion both without and with air resistance, and the computed predictions are compared with the experimental results from Chapter II. The computer is also effective for the analyses of experimental errors.

Chapter IV introduces Newton's second law. With a slight modification of his projectile motion program, the student studies the motion of earth satellites and Apollo spacecraft. These calculations are explicitly used to illustrate Kepler's laws and the laws of conservation of energy and angular momentum.

In Chapter V on student project work, we show how the basic simularity of all Newtonian mechanics problems makes it easy to convert a program from one problem to another. (For example, almost identical computer programs predict the orbit of an Apollo spacecraft and the path of a ball bouncing on a spring.) As an incentive and source of ideas for further project work, we review a variety of student projects.

Chapter VI consists of separate sections that can be inserted into the rest of the course as desired. Section I is the computer analysis of a simple air track experiment. In Section II, the analytic solution of the harmonic oscillator problem is compared with the computer solution. Some calculus is required for this work. The analytic solution in Section III of the damped harmonic oscillator should be challenging even to students with relatively advanced calculus backgrounds. Section IV investigates the motion of a ball bouncing and swinging on the end of a spring. This combination of a strobe experiment and a computer analysis provides a detailed check of Newton's second law.

To order:

send $3.75 (plus postage) to The Dartmouth Bookstore, Main Street, Hanover, New Hampshire 03755. (Postage: $.50/book; $.25/additional book to same destination.)

Orders of more than twenty five (25) copies may be addressed to the Documents Clerk, Kiewit Computation Center, Dartmouth College, Hanover, New Hampshire 03755. Please include payment with order.

APPENDIX C

Working Paper #7

Magnetic Tape Format for Exporting Programs

CONDUIT

Magnetic Tape Format for Exporting Programs

by

Christopher G. Hoogendyk

A consortium of regional networks at
Oregon State University, North Carolina Educational Computing Service,
Dartmouth College, and the Universities of Iowa and Texas (Austin).

Magnetic Tape Format for Exporting Programs

by

Christopher G. Hoogendyk

May 1974

Dartmouth-CONDUIT
Working Paper #7

DRAFT COPY -- not to be reproduced without the written permission of the author.

Magnetic Tape Format for Exporting Programs

This guideline describes a general format to be used for sending computer programs on magnetic tapes.

Following is a brief summary of that format:

1. 7 track
2. 800 bits per inch
3. Odd parity
4. No label, no block serials, and no record control words
5. Fixed length records 80 characters long
6. Block length of 80 characters
7. 6 bit IBMEL character set

This format is suitable for sending documentation, source files for computer programs, and data in card image format. There are two exceptions that can be made to this standard, but they should be made only by prior agreement between the two centers involved in the exchange. The first exception is in cases where additional characters, such as the lower case alphabet, are required for the material. This could be achieved within the standard by using one of the extra characters as an escape character, or the two centers involved might agree on a different tape format based on their own systems and an extended character set. The second exception is when the volume of material is either too small to justify using a tape, or too large to use an unblocked format. If a data deck is small enough to send in an envelope, then cards may be used as long as the card code is the hollerith code used by the IBM model 026 keypunch. (See description at the end of this guideline.) If the material is too large to fit on one or two 2400 foot reels using an unblocked format, then the two centers should make an agreement on a reasonable record size and block size. Just as a guide, the writer was able to fit roughly 30,000 records on a 2400 foot reel using the CONDUIT format. By using a 40 records per block, the same reel held roughly 230,000 records. Based on this example, a data set or collection of programs that is more than about 50 or 60 thousand lines or card images in length (roughly, 1000 printer pages if listed) could be written using blocked records. This should only be taken as a rough indication, and an attempt should be made to follow the standard before deciding to deviate from it.

Each of the conventions of the CONDUIT tape format is discussed individually below. Technical details are fairly standard industry wide and should not require any attention from the programmer's point of view. All technical or hardware specifications should follow the ANSI standard X3.22-1973.

1. Nine track tapes are an innovation of IBM to facilitate the use of their 8-bit EBDIC character set. Most non-IBM systems do not use them and 3 of the 5 CONDUIT centers do not have 9 track tape handlers. Since all of the CONDUIT centers have 7 track handlers, this is the obvious choice.

The ANSI standard X3.22-1973 requires 9 track handlers so that the 7 bits for an ASCII character will fit in one frame. Although this may be fine as a standard to move toward, it is not practical for current use.

2. A density of 800 BPI seems to be a good compromise between the various standard densitites. All the CONDUIT centers can handle it. It was chosen over 556 BPI because some storage capacity will be lost by using unblocked records, and it doesn't make sense to waste storage with all the conventions.

3. Odd parity is used because even parity requires the character code 00 to be mapped to something else to keep it from being confused with blank tape. The code 00 is used for the space or 0 character in different six bit codes, and it is a serious shortcoming for it to be mapped into another character and thereby confused with that other character.

4. The label, block serial number, and record control words are all very different from system to system. Although they add useful error checking for in-house tapes, between different centers they only add garbage and greatly increase the difficulty of reading the tapes.

5. In the absence of a record control word the records have to be some fixed length, and 80 is a logical choice. The 80 character record length is in keeping with the standard card size on card oriented systems. It also allows for the margin width of the large majority of teletypes.

Since standard Fortran only uses columns 1-72 and many systems use columns 73-80 of files or card decks for sequencing numbers, any materials written on tape in the CONDUIT format should only use columns 1-72 and reserve columns 73-80 for sequencing numbers.

6. Since some of the CONDUIT centers have word oriented systems and their word sizes differ, difficulties can arise in separating multiple records within a block. The easiest solution to this problem is simply not to put more than one record in a block. If the tape is treated a full block at a time, and excess at the end of the block is thrown away, then this allows room for some slop at the end of the record to accomodate the differences in machines.

7. Six-bit code can be handled by all the CONDUIT centers, but each has its own code. The Dartmouth system uses 9-bit ASCII internally, but has facilities for loading character tables to translate other codes. One of these other codes is referred to as IBMEL. The two IBM systems both use 8-bit EBCDIC internally, but have facilities for directly reading 6-bit BCDIC from 7 track tapes. It turned out that their 7 track tape BCDIC is the same as the code Dartmouth refers to as IBMEL. Since there are so many codes called BCD, this code will be referred to as IBMEL. The two CDC systems each have different 6-bit BCD codes. Neither of these systems have any facility for reading a tape in a specified format except by writing a program to handle such a job. Since this is the case, it is no additional burden to have a particular character table in that program. Therefore, the best choice is to use the IBMEL character code for transporting materials on magnetic tapes between the CONDUIT centers. A table of the IBMEL code is at the end of this guideline.

Following are some samples of how to work with a tape in the recommended format at each of the CONDUIT centers. These samples are not intended to be exhaustive but should give a fair idea of how to deal with the tapes.

Dartmouth College
Honeywell G635
Dartmouth Time-Sharing System

The G635 is a word oriented machine with 36-bit words. The record lengths specified in the background programs are in terms of words so that a record length of 14 results in an 84 character record on the tape. The resulting tape can be read at another center by specifying 84 character record lengths or by just looking at the first 80 characters of each physical record.

The following background program will write a number of files to a tape:

```
100 WRITE file1;file2;...fileN
110 FORMAT ASCII
120 TO TAPE name-of-tape
130 FORMAT NLABEL;NSER;NRCW
140 FORMAT RECSIZ 14;BLKSIZ 14;FILREC;IBMEL
150 POSITION 1
160 REPORT tape name-of-tape is on the counter
170 END
```

The next background program will read a number of files from a tape:

```
100 READ file1;file2;...fileN
110 FORMAT ASCII
120 FROM TAPE name-of-tape
130 FORMAT NLABEL;NSER;NRCW
140 FORMAT RECSIZ 14;BLKSIZ 14;IBMEL
150 POSITION 1
160 END
```

University of Iowa
IBM 360/65
OS/360

The following deck will print a tape:

```
//jobname  JOB  (xxxxxxxx),‘name'
//  EXEC   PGM=IEBPTPCH
//SYSPRINT  DD  SYSOUT=A
//SYSUT1  DD  UNIT=TP7TRK,LABEL=(1,NL),VOL=SER=tpname,
//  DISP=(OLD,KEEP),
//  DCB=(DEN=2,TRTCH=T,RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT2  DD  SYSOUT=A
//SYSIN  DD  *
    PRINT  MAXFLDS=1
    RECORD  FIELD=(80)
//
```

Notes:

1. To punch a tape use:

```
//SYSUT2  DD  SYSOUT=B
```

2. Use LABEL=(2,NL) for second data set on tape, etc.

The following deck will transfer a card deck to tape:

```
//jobname  JOB  (xxxxxxxx),'name'
//  EXEC   PGM=IEBGENER
//SYSPRINT  DD  SYSOUT=A
//SYSIN  DD  DUMMY
//SYSUT2  DD  UNIT=TP7TRK,LABEL=(1,BLP),DISP=(NEW,KEEP),
//  VOL=SER=tpname
//  DCB=(DEN=2,TRTCH=T,LRECL=80,BLKSIZE=80,RECFM=F)
//SYSUT1  DD  *

     input card deck

/*
```

Notes:

1. Use LABEL=(2,BLP) for second data set, etc.

2. BLP ('bypass label processing') results in a no label tape exactly like NL but BLP, unlike NL, makes no test for the presence of (standard) labels on the tape and thus allows the use of blank tapes or allows overwriting of non-blank tapes. After the tape is created it may be read at NL.

North Carolina Educational Computing Service
IBM 370/165
OS/360 release 21.6

The following deck will read a file from a tape:

```
//jobname  JOB  account,programmer,options
//  EXEC  TAPEDISK
//INPUT  DD  DISP=(OLD,KEEP),UNIT=7TRACK,
//  VOLUME=SER=tapename,DCB=(RECFM=F,
//  BLKSIZE=80,LRECL=80,TRTCH=T),LABEL=( ,NL)
//OUTPUT  DD  DSN=filename,DISP=(NEW,KEEP),
//  UNIT=DISK,VOL=SER=ecs111,SPACE=(TRK,(1,1)),
//  DCB=(LRECL=80,RECFM=FB,BLKSIZE=6400)
//
```

The next deck will read a file from a tape and print it on the printer:

```
//jobname  JOB  account,programmer,options
//  EXEC  TAPEPRNT
//INPUT  DD --repeat DD description as in above deck
//
```

The next deck will read file from a tape and punch it on cards:

```
//jobname  JOB  account,programmer,options
//  EXEC  TAPECARD
//INPUT DD --repeat DD specifications from above decks
//
```

The next deck will write a disc file out to tape:

```
//jobname  JOB  account,programmer,options
//  EXEC  DISKTAPE
//INPUT  DD  DSN=filename,DISP=SHR
//OUTPUT  DD  DISP=(OLD,KEEP),UNIT=7TRACK,
//  VOL=SER=tapename,DCB=(RECFM=F,LRECL=80,
//  BLKSIZE=80,TRTCH=T),LABEL=( ,NL),RING=IN
//
```

The next deck will write a card deck out to tape:

```
     //jobname  JOB  account,programmer,options
     //  EXEC  CARDTAPE
     //OUTPUT  DD --repeat DD specifications from above deck
     //INPUT  DD  *
          ... INSERT CARD DECK HERE
     /*
     //
```

Oregon State University
CDC 3300
OS-3 version 4.3

The following Fortran program will read a number of files from a tape and save them.

```
      PROGRAM READ
      CHARACTER IN(80)
      INTEGER TABLE(64),OUT(20)
      DATA ((TABLE(J),J=1,64)=48,1,2,3,4,5,6,7,
     1  8,9,0,29,46,12,11,30,10,49,50,51,52,53,
     2 54,55,56,57,63,59,14,63,47,63,32,33,34,
     3 35,36,37,38,39,40,41,63,43,44,28,31,63,
     4 13,17,18,19,20,21,22,23,24,25,63,27,26,
     5 60,16,42)
      EQUIVALENCE (IN,OUT)
C     FILE NUMBER 1 = MAG TAPE
C     FILE NUMBER 2 = FILE
      N=FFIN(60)
      DO 100 II=1,N
      READ (60,1000) FILENAME
1000  FORMAT(A8)
      CALL EQUIP(2,8HFILE    )
25    BUFFERIN (1,1) (OUT(1),OUT(20))
      IF (EOF(1)) GO TO 75
      LEN=4*LENGTHF(1)
      DO 50 I=1,LEN
         J1=IN(I)+1
        J2=TABLE(J1)
         IN(I)=J2
50    CONTINUE
      WRITE (2,1001) (IN(J),J=1,LEN)
1001  FORMAT(80R1)
      GO TO 25
75    CALL SAVE(2,FILENAME)
100   CALL UNEQUIP(2)
      CALL EXIT
      END
```

The above program could be run in batch mode using the following job file (it assumes the program is saved with the name READ):

```
[JOB,Job-number,password,message
[EQUIP,1=MT tape-number AT 800 BPI NO RING message
[FORTRAN,I=READ,L,R
8
EPOP
EXPOP
```

```
SIGPOP
RANDK
COMPET1
COMPET2
COMPET5
TUNDRA
[LOGOFF
```

The next program will write a number of files to a tape. It can be run using a job file similar to the first.

```
      PROGRAM WRITE
      CHARACTER IN(80)
      INTEGER TABLE(64),OUT(20)
      DATA ((TABLE(J),J=1,64)=10,1,2,3,4,5,6,7,8,9,16,
     1 14,13,48,28,31,62,49,50,51,52,53,54,55,56,57,
     2 60,59,45,11,15,46,32,33,34,35,36,37,38,39,40,
     3 41,63,43,44,31,12,30,0,17,18,19,20,21,22,23,
     4 24,25,31,27,61,31,29,31)
      EQUIVALENCE (IN,OUT)
C
C     THIS PROGRAM IS DESIGNED TO READ A FILE FROM THE
C     CDC 3300 (OS-3), MAKE THE NECESSARY CHARACTER
C     CONVERSIONS, AND WRITE IT ON A MAG TAPE
C
C     THE ARRAY "TABLE" IS USED TO CONVERT "ORGBCD" TO "IBMEL"
C     WITH THE FOLLOWING EXCEPTIONS
C     1 LEFT BRACKET TO QUESTION MARK
C     2 UP ARROW     TO QUESTION MARK
C     3 RIGHT BRACKET TO QUESTION MARK
C     4 BACK SLASH   TO QUESTION MARK
C
C     THE LINE PRINTER CHARACTER SET IS NOT THE SAME
C     AS THE INTERNAL "ORGBCD" CHARACTER SET.
C
C     FILES ON THE MT ARE SEPARATED BY EOF'S
C     THE END OF THE MT IS INDICATED BY 2 EOF'S
C
      N=FFIN(60)
      DO 100 II=1,N
      READ(60,1000) FILENAME
1000  FORMAT(A8)
      CALL EQUIP(1,FILENAME)
25    READ(1,1001) (IN(J),J=1,80)
      IF(EOF(1)) GO TO 75
1001  FORMAT(80A1)
      DO 50 I=1,80
         J1=IN(I)+1
         J2=TABLE(J1)
         IN(I)=J2
```

```
50    CONTINUE
      BUFFEROUT (2,1)  (OUT(1),OUT(20))
      GO TO 25
75    CALL UNEQUIP(1)
100   ENDFILE 2
      ENDFILE 2
      STOP
      END
```

The University of Texas at Austin
CDC 6400 and CDC 6600
UT2-D

Working with CONDUIT tapes at Texas is similar to working with them at Oregon State. A Fortran program can be written to do the job. They have their own routines for dealing with character data and use them to spread the characters out in an integer array and then pack them back down into the output array. The same problem was solved at Oregon by equivalencing an integer and character array. The programs are not listed here because the reader should be able to get the idea from the Oregon State examples and in any case will need to write a program of his own using these ideas.

Tape Terminology

This description of terms comes from a handout at the University of Texas Computer Center, from ANSI standard X3.22-1973, and from common knowledge among programmers. It does not get into technical specifications and measurements on the grounds that these are fairly standard industry-wide and the average programmer has no need for those details. Anyone who is interested can refer to the ANSI standards.

Magnetic Tape. Magnetic tape is a uniform material one-half inch wide coated on one side with metallic oxide that can be magnetized to record data. The only distinction between nine track and seven track tapes is the handler that wrote the tape. What this means is that a "nine track" tape could be erased and then used as a "seven track" tape.

Block (Physical Record). A block is a group of characters that is treated as a unit by the tape handler. It is written on the tape as a continuous stream of characters with a gap of blank tape before and after it.

Interblock gap.(Inter-record gap) An interblock gap is the section of blank tape that separates recorded blocks. This gap is necessary for the tape handler to stop and get up to speed between blocks.

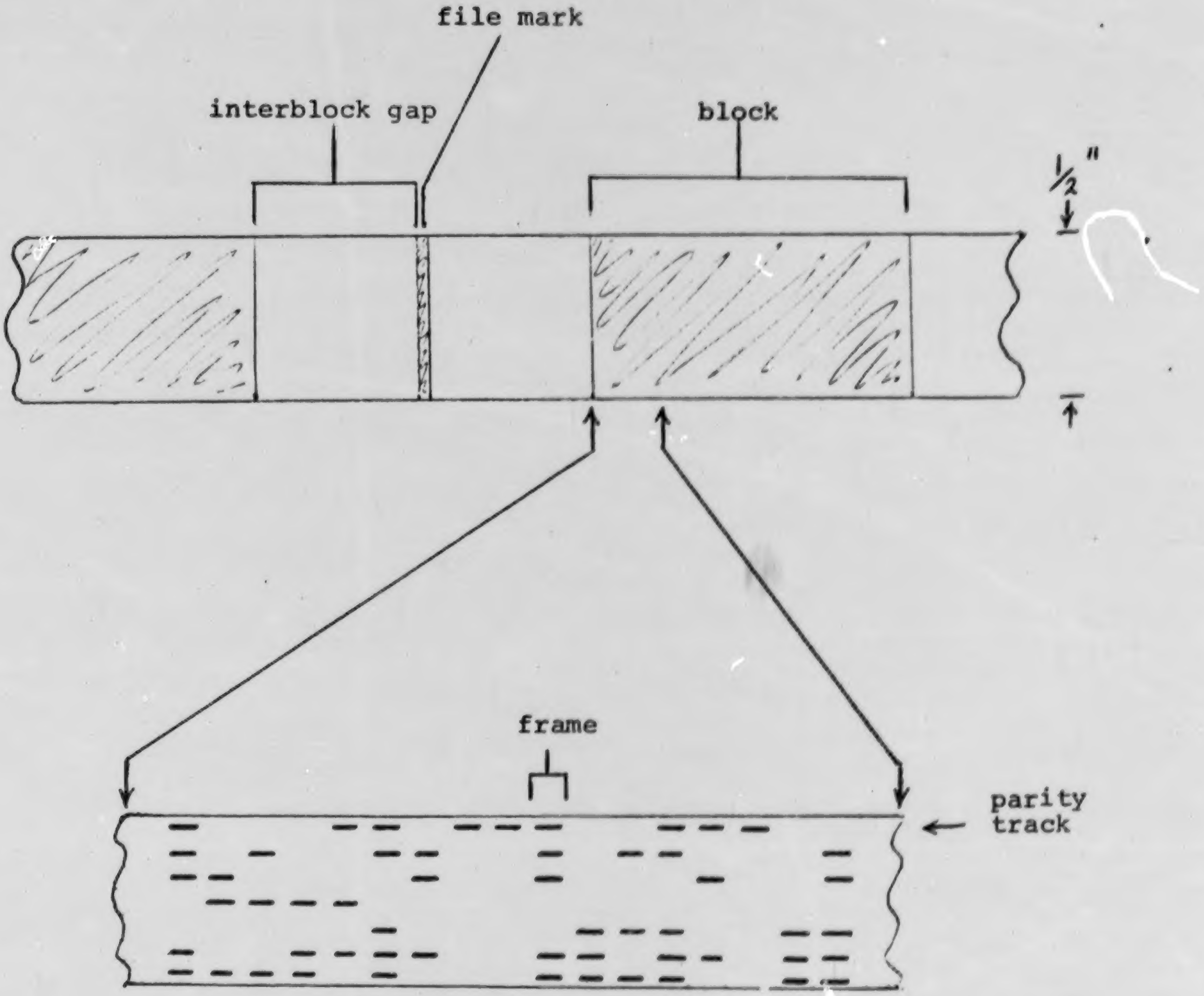
Logical Record. A logical record is a conceptual subunit of a block. There may be one or more logical records per block. A logical record can be a card image, a line in a teletype file, the entire block, etc..

Frame. A frame is one bit position across the width of the tape. Since the tape is written with seven tracks, one frame contains seven bits of information - six bits of data and one parity bit.

Odd Parity. Of the seven bits in a frame six are used for actual data. The seventh is used for a parity check. When the frame is written, the bits in the frame are counted. If the sum of the bits is even, the parity bit is turned on. If the sum of the bits is odd, the parity bit is left off. When the frame is read back the sum of the seven bits will always be odd. If the sum is not odd, it causes a parity error. This warns the programmer that the tape is not recording properly.

File Mark. A file mark is a block made up of a single character. On any given system the same character code is always used, but this differs sometimes and most systems will recognize any single character block as a file mark. These are used to separate different files (programs, data sets, whatever) on a tape.

SECTION OF A RECORDED MAGNETIC TAPE


file mark
interblock gap
block
½"
frame
parity track

Note: these drawings are not drawn to scale

CONDUIT *TAPE DISTRIBUTION FORM*

1. Source (name of center)	2. Computer	3. Operating System	4. Word Size	Date Sent
TO:				Date Received

5. Tracks
*☐ 7 track
☐ 9 track

6. Density
*☐ 800 BPI
☐ Other ____________

7. Parity
*☐ Odd
☐ Even

8. Encoding: a) Character set
*☐ IBMEL (see back)
☐ Other ____________________
(If other, attach listing of character code)

b) Bit size of characters
*☐ 6 bit
☐ Other ____________________

9. Physical blocksize (in characters)
*☐ 80
☐ Other ____________________

10. Record Size (in characters)
*☐ 80
☐ Other ____________________

11. Number of data sets or files
____

12. Additional information
*☐ none
☐ record control words
☐ end of record (line) character(s)
☐ block serials
☐ label
☐ Other ____________________

Names and lengths (in blocks) of files or data sets:

Documentation available: ☐ Abstract ☐ Educational Documentation ☐ Technical Documentation ☐ Local Use Description ☐ Reviews

Other ____

If you have questions about this tape, contact: Name ______________________ Phone ______________

Address ____

Prepared by: ______________________ Authorized by: ______________________

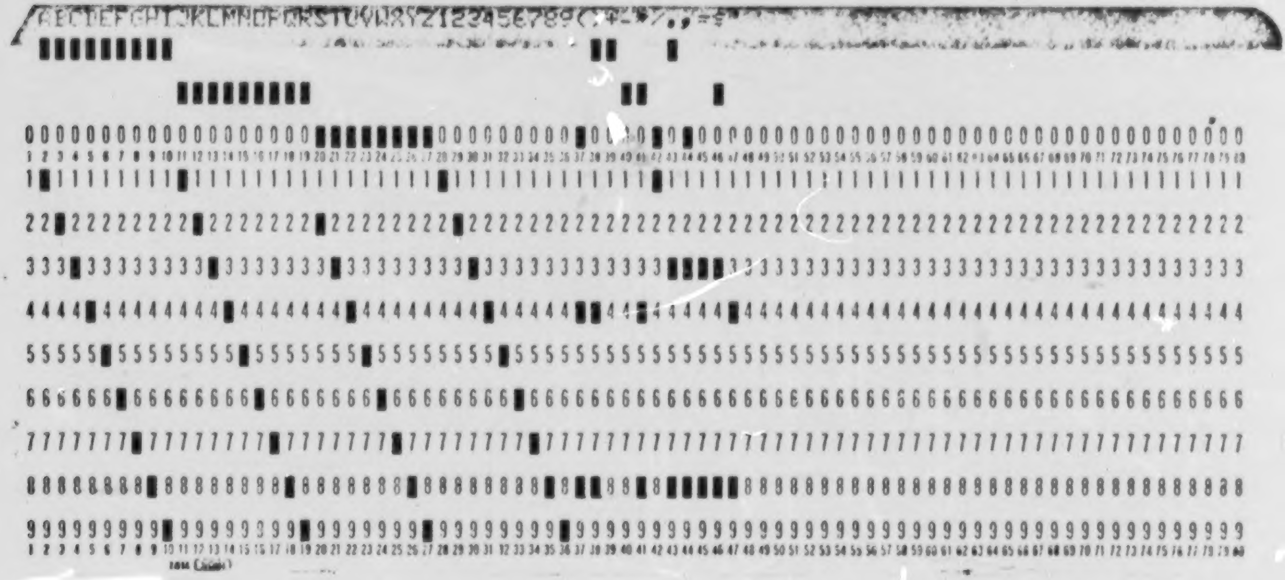
1/10/74 *denotes CONDUIT standard

IBMEL Character Set
(referred to by IBM as 7-track Tape BCDIC)

DEC.	OCT.	HEX.	CHAR.	DEC.	OCT.	HEX.	CHAR.
0	00	00	SP	32	40	20	-
1	01	01	1	33	41	21	J
2	02	02	2	34	42	22	K
3	03	03	3	35	43	23	L
4	04	04	4	36	44	24	M
5	05	05	5	37	45	25	N
6	06	06	6	38	46	26	O
7	07	07	7	39	47	27	P
8	10	08	8	40	50	28	Q
9	11	09	9	41	51	29	R
10	12	0A	0	42	52	2A	(XT)
11	13	0B	Ø	43	53	2B	$
12	14	0C	@	44	54	2C	*
13	15	0D	'	45	55	2D	)
14	16	0E	=	46	56	2E	;
15	17	0F	"	47	57	2F	(XT)
16	20	10	:	48	60	30	&
17	21	11	/	49	61	31	A
18	22	12	S	50	62	32	B
19	23	13	T	51	63	33	C
20	24	14	U	52	64	34	D
21	25	15	V	53	65	35	E
22	26	16	W	54	66	36	F
23	27	17	X	55	67	37	G
24	30	18	Y	56	70	38	H
25	31	19	Z	57	71	39	I
26	32	1A	(XT)	58	72	3A	(XT)
27	33	1B	,	59	73	3B	.
28	34	1C	%	60	74	3C	<
29	35	1D	_	61	75	3D	(
30	36	1E	>	62	76	3E	+
31	37	1F	?	63	77	3F	!

(XT) - Extra character should be avoided unless mapped by previous agreement.

Card decks should use the model 026 IBM keypunch hollerith code. The easiest way to check this is to punch a card with all the characters on it and compare it with the card image here


ABCDEFGHIJKLMNOPQRSTUVWXYZ123456789()+-*/.,=$

APPENDIX D

Working Paper #1

Preparing a Program for Export

CONDUIT

Preparing a Program for Export

by

Christopher G. Hoogendyk

A consortium of regional networks at
Oregon State University, North Carolina Educational Computing Service,
Dartmouth College, and the Universities of Iowa and Texas (Austin).

Preparing a Program for Export

by

Christopher G. Hoogendyk

April 1974

Dartmouth-CONDUIT
Working Paper #1

DRAFT COPY -- not to be reproduced without the written permission of the author.

PREPARING A PROGRAM FOR EXPORT

Introduction

Moving a program successfully involves preparing the program for export, preparing the documentation for the program, shipping the source code with complete documentation, implementing the program at the receiver's site, and verifying that the implemented program provides valid results. The first stage, preparing a program for export, means structuring, coding, and documenting a problem in such a way that the programmed solution can be easily transferred to other computers, documented and verified. It requires a continual awareness of the problems involved in the latter stages of the transfer and a questioning of usual programming assumptions and practices since foreign computer environments may differ in ways surprising to the unsuspecting programmer. These guidelines deal with practices and procedures for the preparation of a program that make the entire transfer run more smoothly.

Before deciding to export a program, however, it is reasonable to ask whether that program ought to be exported. Evaluation for possible program transfer involves comparing system characteristics such as machine speed, accuracy, core space, and special system features (e.g. graphing capabilities). Real differences on this level may make transfer impractical or undesirable.

General

A primary consideration in preparing a program is readability. A programmer other than the one who prepared the program should be able to pick it up, read it, and understand it as quickly and easily as possible. This consideration is inherent in most of the practices and procedures described in the following sections. Programs must be written in a standard applications language such as Fortran or Basic, and simple algorithms are preferrable to cute or tricky ones unless efficiency dictates otherwise. A program should be thoroughly commented and well organized in a structured or modular manner whenever possible. Readability is important not only when exchanging programs with other systems but also when programs are transferred to or maintained by other programmers on the same system.

Another important consideration is the ease with which a program can be implemented on another system. This militates against assembly language and non-standard language features or system routines. To meet this problem, a programmer cannot rely soley on his own system's manuals. He must read and refer to standard specifications of the language and even look through manuals from other systems.

Finally, the ease with which the transferred program can be debugged and verified must be considered. This requires such things as specifying and checking allowed ranges for input parameters, specifying required accuracy where it is critical, and again proper program organization.

The above three considerations apply both to programs that are being written for the first time and to programs that have already been written and which require modification to meet these guidelines. The following sections will deal with specific programming practices.

Program Preparation

The following discussion of programming practices is divided into six basic ideas:

1. Use of a standard language
2. Modular Programming
3. Structured Programming
4. Parameterization
5. Ease of verification
6. Documentation

At a minimum, a program that is going to be transferred should meet these guidelines for language, recognition of machine dependent and external routines (under modular programming), complete test materials (under verification), and documentation. Whenever possible, these guidelines should be met in every respect. Furthermore, anyone developing new programs which will be transferred should follow these guidelines in every respect.

1. Use of a Standard Language

Using a standard language such as ANSI Fortran or Basic.increases the range of potential transfer of a program. A program which is written in assembly language, or in a locally developed applications language, obviously cannot be of much use to other computer installation. Even when a standard language is used, the programmer has to be careful not to rely solely on his own center's documentation. Knowing what is standard and what features are system specific depends on reading and referring to standard language specifications (ANSI documents or the multitude of beginners manuals by McCracken) and reading manuals from different kinds of computer installations and comparing them with the local documentation. If a program requires the use of a nonstandard feature, then that code should be isolated in a module and its purpose explained in internal program comments.

2. Modular Programming

Programs should be written in modular form based on function and machine dependancy. Proper program organization can be achieved most easily by planning ahead and flowcharting the problem at a high functional level. After this has been done coding the problem becomes much easier.

There are several reasons for writing programs modularly. If machine dependent code is isolated in a module whose function is clearly described in comment lines, that module can be more easily replaced by an equivalent module at another installation. This reduces the amount of time spent recoding and simplifies the task of debugging and verifying the program when it is transferred.

The primary core in which a program runs can be utilized with or without overlay techniques depending on local resources and system design. Typically, a batch system with lots of core can run very large programs, while a time-sharing system is much more limited in the amount of core any given program can use and needs to overlay subprograms to fit a large program into its allotted core. If a program is written using subprograms as modules and doesn't have any excessively large pieces of code that are not broken into modules, then it should be able to run in either of these environments without requiring major redesign and recoding of the program. It should be noted here that program controlled overlay is not standard and should be explained in comment lines if it is used.

Another advantage of modular programming is that if a programmer uses standard functional modules in his programs he reduces the amount of time required to develop the program. When the program is transferred implementation time is reduced since the receiving site can substitute their own modules where necessary reducing the amount of unique code that has to be deciphered and debugged.

Modular programming also increases the readability of a program and reduces the time required for someone else to decipher and understand the code.

3. Structured Programming

Within each module the logic flow should be organized in a simple, coherent manner. This means that a page of code should not be a maze of GOTO's with various lines of code inserted as an after-thought. Structured programming also refers to the visual layout of the page. In this sense it means using meaningful variable names and utilizing the width of the page through indentation so that the program can be read more easily.

4. Parameterization

Parameterization should be used to eliminate explicit references to physical devices, character codes, and other system dependent features. Meaningful variable names should be used in the place of recurring constants. These variables should be assigned values at the beginning of the program and comment lines at that point should describe their purpose. This makes program transfer or modification easier. In the case of device numbers, for example, only one line needs to be changed rather than locating all the READ's and WRITE's and replacing them.

5. Ease of Verification

There are several ways in which the problems of verification can be made easier. Besides following all of the above considerations, input data should be checked and informative error messages printed out if they are out of range or in any way unreasonable. Limitations on input parameters and expected accuracy for all possible data should be explained in accompanying documentation as well. When a program is transferred, all necessary documentation for testing should be sent with it. This includes a description of all program options, input data for all paths and options, output for all sample input, and a listing and explanation of all program generated messages. Testing at the receiver site should include error paths and those extreme cases where machine accuracy might degrade either due to computational techniques or due to word size incompatibilities.

6. Internal Program Documentation

Documentation is an important part of program preparation and a necessity for the smooth transfer of programs. All of the information in the following sections should be included as comment lines in the program itself. For a description of additional documentation that should accompany a program when it is transferred, refer to "Describing a Program for Export."

a. Program Header

There should be a standard header of comment lines at the beginning of the program. This header should include:

- i. the mnemonic or calling name of the program,
- ii. a description of the program's origin and subsequent modifications stating author, date, language, machine, and operating system for each version,
- iii. a short abstract describing the function or purpose of the program and its output,
- iv. instructions on how to use the program and any options offered by the program,
- v. a description of the important variables used in the program. This can appear as a table of the variable names with a brief description of the functions they serve in the program. Either part iii, iv, or v of the header should include a description of the allowed ranges for the input parameters.

vi. an explanation of any subroutines, system routines, data sets, or files required by the program that are not a part of the source deck itself. This section should include a description of any interfaces with other programs or routines and of the format of any data sets or files used.

b. Module Header

There should be a standard header of comments at the beginning of every module. This header is a brief form of the program header and includes the following items:

i. the mnemonic or calling name of the module,
ii. a description of its origin and subsequent modifications if that description is different from the rest of the program (this would apply to machine dependent modules that have to be completely rewritten at each installation),
iii. a short abstract describing the function or purpose of the module and if it is machine dependent an indication that it is with any necessary explanation of how to convert it,
iv. a description of the module's interface with the main program indicating the entrance and exit conditions and parameters.

c. Instruction Comments

Comments within the program and all modules should describe any statements that deviate from the language standard. They should also be used to describe the function of any instructions, algorithms or equations involving unusual coding or complex logic.

d. Program Logic

Comments within the program and all modules should describe the organization and flow of logic through the program. These comments should be designed so that a programmer who is unfamiliar with the program can read through it quickly and easily and be able to understand what the program is doing.

Checklist of Things to Watch For

I. Machine Oriented Items.

1. Does the new machine have the language in which the program is written? Is their version of the language standard? Are all the necessary functions and library routines available?

2. Is the core size of the new machine adequate?

3. Are word and byte size of both machines the same? If not, will it make a difference ? Is the precision of the new machine adequate?

4. Does the new machine have tape handlers and the ability to read the source deck from an outside magnetic tape? Does the program require the use of devices (e.g. plotters) that may be unavailable?

5. Will the speed of the new machine be adequate to run the program efficiently?

6. Does the program require the use of characters that may not be available on the new machine? For example, a machine that uses a 6-bit code will not have lower case alphabetics.

II. Program Oriented Items

1.. Has the program been written with consideration for these guidelines?

2. Has the program been thoroughly and successfully tested on your machine?

3. Have all variables been initialized in the program ? This should be done even if your current system does it for you.

4. Are you sure that the correct source and documentation are sent together? Are they up to date versions that agree with what you are running on your system?

5. Are non-integer numeric comparisons made using range of magnitude rather than absolute matches? For example, if you assign A the value of 1.-(1.5+.5)/2.0 it may come out as 0.0000001 and a subsequent test to see if A is equal to zero would fail! An alternative would be to test the absolute value of A to see if it is less than 0.000001 or whatever range is suitable.

6. If you depend on any internal representation of your machine its use should be isolated to one module and coded in a way that is clear and easy to modify. For example, if you are using a character code, do it through a lookup table.

7. Avoid any condition or construction that is left undefined by the standard for the language you are using. For example, in Fortran the expression A**B**C is undefined and may be interpreted as either A**(B**C) or (A**B)**C. In this case put in the parenthesis to specify exactly what you mean. Another example is when the argument of a computed GOTO in Fortran is less than 1 or greater than the number of statement numbers given. In this case the computed GOTO should be preceded by a conditional to handle those cases if they occur.

```
100 REM  PROGRAMX -- (BASIC PROGRAM STARTS AT LINE 720)
110 REM
120 REM  VERSION 1:  WRITTEN BY PROF. D. DUCK, UNIVERSITY OF QWERTY,
130 REM       QWERTYVILLE, USA  09090.  LAST MODIFIED DECEMBER 4, 1971.
140 REM       WRITTEN IN FORTRAN LEVEL G FOR THE IBM 360, OS-4.3.
150 REM
160 REM  VERSION 2:  MODIFIED BY ED HORSE, COWTOWN, USA  80808.
170 REM      RAN IN FORTRAN VERS. 21.5 FOR THE CDC 6400.  LAST
180 REM      MODIFIED APRIL 16, 1972.
190 REM
200 REM  VERSION 3:  TRANSLATED BY M. MOUSE, DARTMOUTH COLLEGE, HANOVER
210 REM       N.H.  03755.  RUNS IN DARTMOUTH BASIC SIXTH EDITION ON
220 REM       A HONEYWELL G635, DARTMOUTH TIME-SHARING SYSTEM.
230 REM       USES 8192 WORDS OF 36 BIT CORE AND 2 SECONDS OF CPU TIME.
240 REM
250 REM  THIS IS VERSION 3.  LAST MODIFIED JANUARY 13, 1973.
260 REM
270 REM
280 REM  DESCRIPTION:
290 REM
300 REM       HERE IS A CONCISE DESCRIPTION OF THE PROGRAMS FUNCTION
310 REM  PURPOSE, AND OUTPUT.
320 REM
330 REM
340 REM  INSTRUCTIONS:
350 REM
360 REM       HERE IS THE INFORMATION NECESSARY FOR SOMEONE TO USE
370 REM  THE PROGRAM.  THIS CAN EITHER BE A COMPLETE DESCRIPTION OF
380 REM  HOW TO USE THE PROGRAM, A REFERRENCE TO A MANUAL AND WHERE
390 REM  IT CAN BE GOTTEN, OR JUST A NOTE SAYING THAT INSTRUCTIONS ARE
400 REM  GIVEN WHEN THE PROGRAM IS RUN.
410 REM
420 REM
430 REM  PROGRAM NOTES:
440 REM
450 REM                    TABLE OF IMPORTANT VARIABLES
460 REM                    ****************************
470 REM
480 REM          TABLE(50)  --  TABLE OF ...
490 REM          TOP        --  POINTER TO TOP OF TABLE
500 REM          IND1       --  INDICATOR FOR ...
510 REM          INPUT(10)  --  USER INPUT PARAMETERS
520 REM            .
530 REM            .
540 REM          ETC.
550 REM
560 REM     THIS PROGRAM USES A DATA FILE THAT IS A RANDOM ACCESS NUMERIC
570 REM  FILE.  SUCH A FILE IS SIMILAR TO A DECLARED ARRAY IN THE PROGRAM
580 REM  BODY IN THAT THE NUMBERS STORED IN IT CAN BE ACCESSED FROM ANY
590 REM  POSITION WITHIN THE LIST.  HOWEVER, A FILE IS USED BECAUSE THE
600 REM  NUMBER OF ENTRIES IS TOO LARGE TO FIT IN CORE AT ONE TIME.  IT
610 REM  IS NECESSARY TO HAVE SOME STRUCTURE THAT CAN CONTAIN UP TO
620 REM  20000 NUMBERS AND FROM WHICH THOSE NUMBERS CAN BE ACCESSED AT
630 REM  RANDOM.  THE FILE IS NUMBERED FROM 0 TO N-1, WHERE N IS THE
640 REM  NUMBER OF ENTRIES.
650 REM
660 REM       ACCURACY SHOULD BE NO PROBLEM SINCE INPUT PARAMETERS ARE
670 REM  INTEGERS THAT RANGE UP TO ABOUT 20000, AND THERE ARE NO
680 REM  CRITICAL CALCULATIONS THAT MIGHT DECAY.
690 REM
700 REM  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
710
```

APPENDIX E

Working Paper #2

Describing a Program for Export

CONDUIT

Describing a Program for Export

by

John M. Nevison

A consortium of regional networks at
Oregon State University, North Carolina Educational Computing Service,
Dartmouth College, and the Universities of Iowa and Texas (Austin).

Describing a Program for Export

by

John M. Nevison

April 1974

Dartmouth-CONDUIT
Working Paper #2

DRAFT COPY -- not to be reproduced without the written permission of the author.

Describing a Program For Export

In an author's own class, his text may make even simple program documentation unnecessary. Similarly, on it's original computer, a well-annotated interactive program may require no explanation beyond how to begin. But often a gap exists between the well documented program and the author's description of how to use it in a course.

As soon as a program is moved from one computer to another, this gap widens. Several new questions confront the programmer who is importing the new program: Where is the program from? What does it do? How can one be sure it works correctly?

To answer these questions a small file entitled Mechanics of Use will be enclosed with each program or set of programs exported from the CONDUIT-Dartmouth office. The actual file name will be the program name followed by the letter Q. For example, PROGNM would have a file on its use named PROGNMQ.

The file, Mechanics of Use, should be written after the programmer has completed his internal program documentation and has briefly reviewed the author's write-up on how the programs are used in a course.

The goal of Mechanics of Use is to span the gap from the program to the text so that a programmer at a remote site can make the program usable in the course.

The file should contain:

1. PROGNMQ
2. Title, "Mechanics of Use For: ..."
3. The name of the author(s) of the file.
4. The date last modified and by whom.
6. A full reference to the text which the program accompanies, (with cost and how to order)
7. A full list of the programs and subprograms.
8. A list of the current locations where the programs work, (Language, machine, place, person)
9. A graceful, intelligent discussion designed to ensure that the recipient can make the program work in the course for which they were intended.
10. Sample input and output, data or special tests necessary to illustrate the discussion.

APPENDIX A

EXAMPLE

```
PLOT0


     *********************************************************
     *  COPYRIGHT 1974 BY THE TRUSTEES OF DARTMOUTH COLLEGE  *
     *********************************************************


    MECHANICS OF USE FOR: PLOT AND THE OTHER PROGRAMS IN THE HUGGINS
TEXT.

   DESCRIPTION WAS ORIGINALLY WRITTEN BY ELISHA HUGGINS.
IT WAS MODIFIED BY DAVID COOLEY ON JANUARY 23,1974.

    LAST MODIFICATION BY JOHN M. NEVISON ON 8 APRIL 1974.

    TEXT:  ELISHA HUGGINS, COMPUTER ASSISTED INTRODUCTION TO MECHANICS,
           COPYRIGHT, TRUSTEES OF DARTMOUTH COLLEGE, 272 PP.,1972.
           COST:  $3.60 FROM DOCUMENT CENTER, KIEWIT COMPUTATION CENTER,
           DARTMOUTH COLLEGE, HANOVER, NH  03755.

      *        *        *        *        *        *        *        *

    BECAUSE OF SPACE LIMITATIONS THE HEADER FOR THE PROGRAM COULD
NOT BE INCLUDED WITHIN THE PROGRAM AND IS PRESENTED HERE:

  PLOT

  VERSION 1:  WRITTEN BY ELISHA HUGGINS, PHYSICS DEPARTMENT, DARTMOUTH
      COLLEGE, HANOVER, NH 03755.
      MODIFIED FOR PROJECT CONDUIT BY DAVID COOLEY JANUARY 1974.

  THIS IS VERSION 1.  LAST MODIFIED BY JOHN M. NEVISON 8 APRIL 1974.

  DESCRIPTION:  THIS PROGRAM IS THE GENERAL UTILITY PROGRAM THAT
      PLOTS ON A TELETYPE THE RESULTS OF PROGRAMS WRITTEN BEGINNING
      WITH A LINE NUMBER GREATER THAN 100.  THE FORMAT OF THESE
      PROGRAMS IS DISCUSSED IN DETAIL IN THE HUGGINS TEXT.

  INSTRUCTIONS:  CONTAINED IN MECHANICS OF USE AND IN TEXT.


      *        *        *        *        *        *        *        *        *


     THE FOLLOWING DISCUSSION IS ADDRESSED TO THE PROGRAM "PLOT"
WHICH APPEARS ON PAGE 5 OF THE TEXT.  THE REMAINING PROGRAMS ARE
AVAILABLE IN THE TEXT AND ARE INTENDED TO BE WRITTEN BY THE STUDENT.


    PLOT IS A PROGRAM USED THROUGHOUT THE TEXT TO PRODUCE STROBE AND
ORBIT PLOTS.  IT IS DESIGNED TO WORK WITH COMPUTERS OF LIMITED STORAGE
CAPACITY.  TO PLOT A MULTI-VALUED FUNCTION, IT IS NECESSARY TO STORE THE
POINTS AS THEY ARE CALCULATED AND THEN PRINT OUT THE PLOT ALL AT ONCE.
THIS IS NECESSARY FOR FUNCTIONS WHICH PRODUCE GRAPHS HAVING BOTH POSI-
```

TIVE AND NEGATIVE SLOPES AS THE TELETYPE CAN NOT ROLL BACK THE PAPER. THIS PROGRAM HAS BEEN GREATLY SIMPLIFIED AND SHORTENED AS COMPARED WITH THE ORIGINAL TO REDUCE THE STORAGE REQUIREMENTS AND MAKE IT COMPATIBLE WITH THE MOST ELEMENTARY FORMS OF "BASIC". AS A RESULT IT SHOULD RUN ON ANY 8K MINI COMPUTER.

"PLOT" HAS BEEN WRITTEN ON LINES ONE THROUGH NINTY-EIGHT AND IS USED AS SUCH IN THE USER'S PROGRAM. THE USER'S FUNCTION PRODUCING PROGRAM BEGINS AT LINE 100 AND JUMPS BACK TO THE PLOTTING PROGRAM TO PLOT THE GRAPH. IN THIS WAY THE PROGRAM CAN BE USED ON ANY COMPUTER HAVING BASIC AND IT DOES NOT RELY ON SPECIAL SUBROUTINING CAPABILITIES. WHEN WORKING WITH LARGE NUMBERS OF STUDENTS ON A MINI COMPUTER, IT MAY BE CONVENIENT TO HAVE THE "SCRATCH" COMMAND MODIFIED SO THAT IT WILL NOT ERASE THESE FIRST 98 LINES. THEN THE PLOTTING PROGRAM CAN BE LEFT IN THE MACHINE FOR REPEATED USE BY STUDENTS.

THERE ARE SEVERAL MODIFICATIONS OF THE PROGRAM WHICH THE INSTRUCTOR MAY WISH TO MAKE. IF A LARGER COMPUTER IS USED, IT MAY BE CONVENIENT TO CHANGE THE PROGRAM TO A SUBROUTINE TO TAKE ADVANTAGE OF THE MACHINE'S SUBROUTINING CAPABILITIES. SOMETIMES A PRINTING ERROR MAY OCCUR DURING THE PLOT BECAUSE A PARTICULAR TERMINAL WILL NOT HANDLE A FULL 72 CHARACTER LINE. THIS ERROR CAN BE ELIMINATED BY CHANGING THE VARIABLES WHICH CONTROL THE PHYSICAL SIZE OF THE PLOT. ALSO THE USER MAY WISH TO CHANGE THE PRINTING CHARACTERS OR THE LABELING OF THE AXES. THESE MODIFICATIONS WILL BE BRIEFLY DESCRIBED; ANY OTHERS WILL BE LEFT UP TO TO THE USER.

"PLOT" AS A SUBROUTINE

IF THE PLOTTING PROGRAM IS TO BE TREATED AS A SUBROUTINE, THERE ARE THREE DISTINCT PIECES WHICH CAN BE TREATED AS THREE SEPARATE SUBROUTINES. THE FIRST, WHICH CORRESPONDS TO "GOSUB 10" SHOULD CONTAIN LINES 2-18 WITH LINE 8 DELETED. THIS SECTION INITIALIZES VARIABLES, CLEARES THE STORAGE VECTOR, AND CALCULATES SCALE FACTORS. THE SECOND, CORRESPONDING TO "GOSUB 20" SHOULD CONTAIN LINES 19-27 FOLLOWED BY LINES 51-64. THIS SUBROUTINE WILL STORE THE POINTS AS THEY ARE CALCULATED. THE THIRD, CORRESPONDING TO "GOSUB 30" SHOULD CONTAIN LINES 30-48 FOLLOWED BY LINES 66-96. THIS SECTION WILL PRINT OUT THE FINAL PLOT.

CHANGING THE PRINT CHARACTERS

THE STANDARD PRINT CHARACTERS IN "PLOT" ARE AN ASTERISK (P1=1) AND AN "OH" (P1=2). THESE CAN BE CHANGED BY REPLACING THE * IN LINE 87 AND THE "OH" IN LINE 89 WITH THE DESIRED CHARACTERS. FOR EXAMPLE, TO HAVE THE COMMAND "LET P1=2" PRINT A PERIOD, LINE 89 WOULD BECOME

```
89 PRINT ".";
```

RELABELING AXES

THE FINAL MODIFICATION DESCRIBED HERE IS THE RELABELING OF THE X AND Y AXES. THESE ARE RELABLED BY REPLACING THE X AND Y IN LINES 41 AND 69 WITH THE DESIRED LETTERS. IT IS ESSENTIAL TO LEAVE THE SPACE IN

FRONT OF THE LETTER IN LINE 69 IN ORDER TO MAINTAIN THE CORRECT LOCATION OF THE VERTICLE AXIS. THE SPACE BEFORE THE LETTER IN LINE 41 SIMPLY LEAVES THE SPACES SEEN BETWEEN THE X'S IN THE PLOTS. THIS CAN BE DELETED IF DESIRED.

TELETYPE COORDINATES

SINCE THE TELETYPE PRINTS 6 LINES PER INCH AND 10 CHARACTERS PER INCH ALONG THE LINE, A 7X7 TELETYPE PLOT CONTAINS 42 LINES OF 70 CHARACTERS EACH. THIS PROGRAM USES THE VARIABLE O1 (OH ONE) TO LABEL THE ROWS FROM ZERO (TOP ROW) 41, AND THE VARIABLE O2 (OH TWO) TO LABEL THE COLUMNS FROM ZERO (LEFT MOST COLUMN) TO 69. THESE VARIABLES (O1 AND O2) IDENTIFY EVERY POINT IN THE PLOT AND FORM WHAT ARE CALLED THE "TELETYPE COORDINATES" OF THE POINT.

BEFORE THE POINTS ARE CALCULATED, THE USER ENTERS THE RANGE OF HIS PLOT AND THE PROGRAM THEN CALCULATES THE SCALE FACTORS REQUIRED TO CONVERT FROM THE USER'S COORDINATES TO THE TELETYPE COORDINATES. THE CALCULATION OF THESE SCALE FACTORS IS THE MAIN PURPOSE OF THE INITIAL "GOSUB 10" PART OF THE PROGRAM.

WHEN THE COMPUTER COMPUTES A POINT, THE "GOSUB 20" PART OF THE PROGRAM FIRST CONVERTS FROM THE USER'S COORDINATE TO THE TELETYPE COORDINATE AND THEN STORES THE POINT IN STORAGE VECTOR O( ). THE GREAT ADVANTAGE OF USING TELETYPE COORDINATES TO STORE THE POINT IS THAT A TELETYPE COORDINATE WILL NOT EXCEED TWO DIGITS, WHILE THE USERS COORDINATES MAY BE EIGHT DIGITS. MANY MORE POINTS CAN BE STORED IN A GIVEN STORAGE VECTOR BY FIRST ROUNDING OFF TO THE CORRECT TELETYPE COORDINATE THAN BY STORING 6 OR 8 FIGURE NUMBERS.

THE PLOTTING PROGRAM CAN PRINT A BLANK, AN ASTERISK, OR AN "OH" AT ANY OF THE 42 X 70 = 2940 POINTS OF A 7X7 PLOT. THIS MEANS IT HAS TO STORE UP TO NEARLY 9000 PIECES OF INFORMATION IN ITS STORAGE VECTOR. YET THE WHOLE PROGRAM INCLUDING THE ROOM SAVED FOR STORAGE SHOULD NOT EXCEED 1000 WORDS IN ORDER TO BE USED ON AN 8K MINI COMPUTER. (TYPICALLY THE BASIC COMPILER USES UP TO 5K WORDS AND THE USER IS LEFT WITH 3K OR LESS.) BY USING A "BASE 3" STORING TECHNIQUE, ALL 9000 BITS OF INFORMATION ARE STORED IN THE VECTOR O( ) WHICH HAS ONLY 210 COMPONENTS. THE "DIM" COMMAND IN LINE THREE SAVES ROOM FOR THIS VECTOR.

BASE 3 STORAGE

ESSENTIAL TO THE STORAGE TECHNIQUE IS THE FACT THAT EVEN THE SMALLEST MINI COMPUTERS USE 32 BITS TO STORE A FLOATING POINT NUMBER (ALL BASIC COMPILERS USE FLOATING POINT NUMBERS). OF THESE 32 BITS, 7 ARE USED FOR THE DECIMAL POINT, 2 FOR THE SIGN AND PARITY, AND 23 ARE LEFT FOR STORING THE NUMBER. THUS ANY NUMBER SMALLER THAN 2**23 IS STORED EXACTLY, WITHOUT ROUNDOFF. THE IDEA IS TO PACK AS MUCH INFORMATION AS POSSIBLE INTO THIS NUMBER. SINCE 3 PIECES OF INFORMATION ABOUT EACH POINT ARE STORED, IT IS IMPORTANT TO KNOW THAT 3**14 IS LESS THAN 2**23, AND THUS IS STORED EXACTLY.

TO ILLUSTRATE THE STORAGE PROCEDURE, SUPPOSE ONE WANTED THE COMPUTER TO PRINT OUT THE FOLLOWING "BLOCK" OF 14 NUMBERS

```
     0  1  2  3  4  5  6  7  8  9  10  11  12  13
    (1  0  0  2  0  0  0  1  0  0   0   0   0   2)
```

LABELING THE POSITIONS OF THESE NUMBERS FROM ZERO TO 13 (FROM LEFT TO RIGHT), WE FIND A "1" IN THE POSITIONS ZERO AND 7, A "2" IN POSITIONS 2 AND 13, AND "0" ELSEWHERE. NOW SUPPOSE THE FOLLOWING NUMBER IS STORED IN THE STORAGE VECTOR:

```
1*3**0 + 2*3**3 + 1*3**7 + 2*3**13
```

SINCE IT IS SMALLER THAN 3**14 IT WILL BE STORED EXACTLY, AND ALL THE INFORMATION NEEDED TO REPRODUCE THE BLOCK OF NUMBERS WILL BE CONTAINED IN IT.

VISUALIZING THE STORED FORM

```
1*3**0 + 0*3**1 + 0*3**2 + 2*3**3 + 0*3**4 + 0*3**5 + ...
```

THE EXPONENT OF THREE GIVES THE POSITION IN THE BLOCK, AND THE COEFFICIENT OF THREE IS THE NUMBER TO BE PRINTED THERE. HOWEVER, INSTEAD OF PRINTING NUMBERS THE PROGRAM COULD PRINT A BLANK FOR 0, A "*" FOR 1, AND AN "CH" FOR 2. THE RESULTS IS:

```
     0  1  2  3  4  5  6  7  8  9  10  11  12  13
    (*        0              *                  0)
```

WHICH OF COURSE COULD BE 14 POINTS IN THE TELETYPE PLOT.

TO USE THIS STORAGE TECHNIQUE IN THE PLOTTING PROGRAM, EACH LINE OF 70 CHARACTERS IS BROKEN DOWN INTO 5 BLOCKS OF 14 POINTS EACH. SINCE A 7 X 7 INCH PLOT HAS 42 LINES, THE ENTIRE PLOT CONTAINS 5*42=210 SUCH BLOCKS. AND, SINCE EACH BLOCK CAN BE STORED AS ONE NUMBER (COMPONENT) OF THE STORAGE VECTOR, A STORAGE VECTOR WITH 210 COMPONENTS IS ALL THAT IS NEEDED.

TELETYPE PLOTTING SUMMARY

A) STEPS

0) START YOUR PROGRAM AT LINE 100 OR LATER.

1) SET THE SCALES OF THE GRAPH BY DEFINING THE VARIABLES (X8,Y8), (X9,Y9), AND (X5;Y5). DEFINE X4 ONLY IF YOU WISH TO LIMIT THE POINTS PLOTTED PER GRAPH. NOW USE THE COMMAND "GOSUB 10" TO INITIALIZE THE PLOTTING SUBROUTINE.

2) DEFINE THE COORDINATES X1, Y1 OF THE POINTER TO BE STORED, STATE THE PLOTTING CHARACTER P1 (1 FOR "*". 2 FOR "O"), AND REPLACE THE "PRINT" COMMAND WITH "GOSUB 20". IF YOU WISH TO HAVE THE VALUES PRINTED OUT PUT THE "GOSUB 20" COMMAND ON EITHER SIDE OF THE "PRINT" COMMAND.

3) PLOT THE STORED POINTS BY ADDING THE COMMAND "GOSUB 30" AFTER THE CALCULATIONS ARE FINISHED. "GOSUB 30" OFTEN APPEARS JUST BEFORE THE "END" STATEMENT.

B) VARIABLES

0) DO NOT USE ANY OF THE O ("OH") VARIABLES SUCH AS O1, O2,...., O9, O( ) IN YOUR PROGRAM. THESE ARE USED BY THE PLOTTING PROGRAM.

1) (X8,Y8) = COORDINATES OF THE LOWER LEFT CORNER OF THE PLOT

2) (X9,Y9) = COORDINATES OF THE UPPER RIGHT CORNER OF THE PLOT

3) (X5,Y5) = PHYSICAL DIMENSIONS, IN INCHES OF THE PLOT. THE PLOT CAN NOT EXCEED 7 X 7 INCHES.

4) P1 = PRINTING VARIABLE. P1=1 PRINTS "*", P1=2 PRINTS "O".

5) X4 = MAXIMUM NUMBER OF POINTS PER PLOT (SEE BELOW)

C) SPECIAL FEATURES

0) TO ERASE STORED POINTS FOR A SECOND PLOT IN ONE PROGRAM, USE "GOSUB 10".

1) TO LIMIT THE NUMBER OF POINTS PER PLOT, USE THE VARIABLE X4 BEFORE "GOSUB 10". IF X4 = 100, THE COMPUTER WILL STORE UP TO 100 POINTS AND THEN PRINT THE PLOT. IT THEN ERASES THE MEMORY AND STARTS STORING THE NEXT 100 POINTS OF THE SAME FUNCTION. THIS ALLOWS A COMPLICATED PLOT TO BE SEPARATED INTO SEVERAL SIMPLER ONES.

FOLLOWING IS A SAMPLE RUN. THE USER CALLS UP THE PROGRAM "PLOT" AND THEN TYPES IN HIS OWN PROGRAM STARTING AT LINE 100. HE THEN RUNS THE PROGRAM TO PRODUCE THE PLOT. THE APPEARANCE OF THE PLOT MAY VARY BETWEEN THE DIFFERENT MAKES OF TERMINALS BECAUSE THE DISTANCE BETWEEN PRINT LINES IS NOT ALWAYS STANDARD. FOR EXAMPLE, ON A "BEEHIVE" TERMINAL THE CIRCLE MAY BE ELONGATED. THE TERMINAL PRINTOUT OF SUCH A USER INTERACTION FOLLOWS:

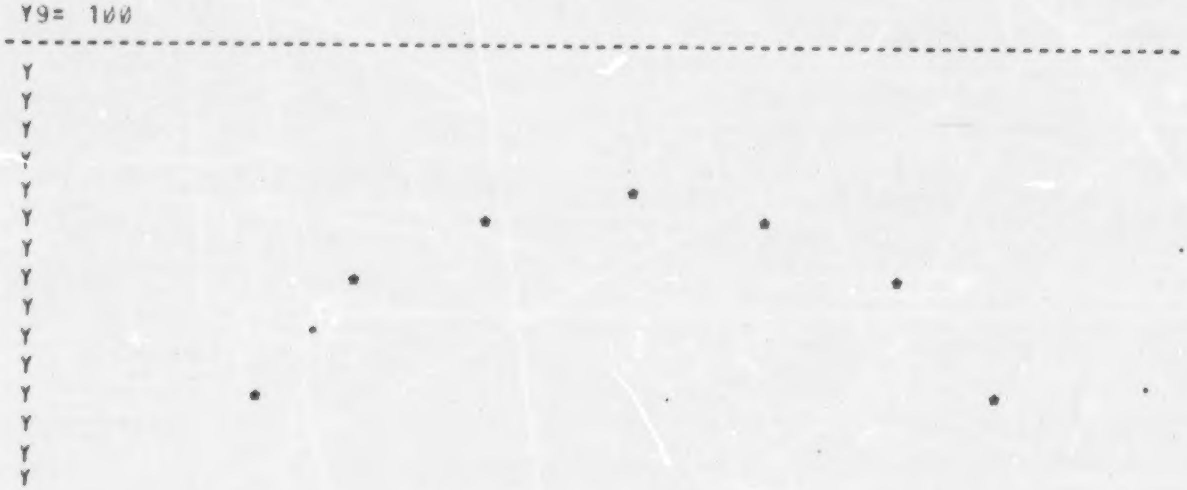
```
OLD PLOT
READY

100 LET P=3.14159
110 LET X8=Y8=0
120 LET X9=Y9=100
130 LET X5=Y5=7
140 GOSUB 10
150 LET R=40
160 LET N=20
170 PRINT "T","X","Y"
180 FOR T=0 TO 2*P STEP 2*P/N
190 LET X=50+R*COS(T)
200 LET Y=50+R*SIN(T)
210 LET X1=X
```

```
220 LET Y1=Y
230 LET P1=1
240 PRINT T,X,Y
250 GOSUB 20
260 NEXT T
270 PRINT
280 GOSUB 30
290 END
RUN


PLOT      01/10/74  12:57


T              X              Y
 0              90             50
 0.314159       88.0423        62.3607
 0.628318       82.3607        73.5114
 0.942477       73.5114        82.3607
 1.25664        62.3607        88.0422
 1.57079        50.0001        90
 1.88495        37.6394        88.0423
 2.19911        26.4887        82.3607
 2.51327        17.6394        73.5115
 2.82743        11.9578        62.3608
 3.14159        10             50.0001
 3.45575        11.9577        37.6394
 3.76991        17.6392        26.4887
 4.08407        26.4885        17.6394
 4.39823        37.6392        11.9578
 4.71238        49.9998        10
 5.02654        62.3605        11.9577
 5.3407         73.5113        17.6392
 5.65486        82.3606        26.4884
 5.96902        88.0422        37.6391
 6.28318        90             49.9998
```



```
Y         *                                                    *
Y
Y
Y
Y
Y        *                                                      *
Y                                                               *
Y
Y
Y
Y
Y         *                                                    *
Y
Y                                                                           *
Y
Y             *                                            *
Y
Y
Y
Y                   *                                *
Y
Y                           *                *
Y                                   *
Y
Y
Y
Y
X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
X8= 0                                                      X9= 100
Y8= 0

0.618 SEC.  31 I/O

    THESE PROGRAMS  WORK IN BASIC AT:

         1. KIEWIT COMPUTATION CENTER
            DARTMOUTH COLLEGE, HANOVER, NH 03755.
            MACHINE: HONEYWELL 635
            CONTACT: JOHN M. NEVISON, CONDUIT OFFICE.

         2. COMPUTER CENTER
            OREGON STATE UNIVERSITY
            CORVALLIS, OREGON  97331
            MACHINE: CONTROL DATA 3300
            CONTACT: JO ANN BAUGHMAN, CONDUIT OFFICE.
```

APPENDIX F

Working Paper #3

Agreeable Fortran

CONDUIT

Agreeable Fortran

by

Christopher G. Hoogendyk

A consortium of regional networks at
Oregon State University, North Carolina Educational Computing Service,
Dartmouth College, and the Universities of Iowa and Texas (Austin).

Agreeable Fortran

by

Christopher G. Hoogendyk

April 1974

Dartmouth-CONDUIT
Working Paper #3

DRAFT COPY -- not to be reproduced without the written permission of the author.

Agreeable Fortran

1.1 Purpose. This document is meant to serve as a supplement to the ANSI specification of Fortran IV. It describes the differences between the existing Fortrans used at the five CONDUIT computer centers, and should be used as a guide for programmers in writing programs that are more easily transported between these five centers. It should be used in close conjunction with the ANSI standard specification of Fortran IV which is available as X3.9-1966 from American National Standards Institute, Inc., 1430 Broadway, New York, New York 10018. The sections in this document are numbered the same as those of the ANSI document. Where all five CONDUIT centers are in agreement with the ANSI document, the section in question does not appear here. Fortran features which are not ANSI standard are not discussed here and should be avoided in practice wherever possible.

The five CONDUIT computer centers are:

	Machine	Operating System	Fortran level
Dartmouth College	Honeywell G635	DTSS	DTSS Fortran
University of Iowa	IBM 360/65	OS/360	IBM Fortran IV
NCECS (TUCC)	IBM 370/165	OS/360 rel. 21.6	IBM Fortran IV
Oregon State University	CDC 3300	OS-3 vers. 4.3	vers. 3.12
University of Texas (Austin)	CDC 6400 & CDC 6600	UT2-D	rel. 60.2

3.1 The ANSI document specifically states that it implies no collating sequence in the order in which it lists the characters used by Fortran. This can be important since in the ASCII character code a numeric code can be checked agains "A" and "Z" to see if it is an alphabetic character, whereas in many BCD codes the alphabet is divided into three separate groups and would require a minimum of six checks. Because of these problems, any use of character codes should be avoided. If it necessary to use them, their use should be described in comment lines in the program. Their use should not involve literal numerics in the program code, but should be mediated through hollerith constants assigned to variables with mnemonic names (DATA PLUS/1H+/) or through the use of a character code lookup table if the use relies on the collating sequence of a particular code. If a lookup table is used, it would be replaced at each center and should be documented as such within the program with a description of the code that it should map into. The program should require no other changes to make use of the new table.

3.2-3.4 Any systems using free format Fortran in which it is not necessary to conform to column numbers on a line should have some means of editing their programs into formatted form adhering to the correct column numbers for the major parts of the line: 1-5 for statement numbers, 6 for the continuation character, and 7-72 for the statement. Programs should be edited into formatted form before export.

4.2.5 Some implementations of Fortran do not have logical data types. Because of this limitation, use of logical variables should be avoided and logical expressions should be used only in logical IF statements.

4.2.6 Hollerith data should be avoided since it is more machine dependent than any other data type and it has no standard implementation. If such data is necessary, it should be used sparingly and its purpose described within the program. See also section 3.1 of this document.

5.1.1.5 See section 4.2.5 of this document concerning logical data types.

5.1.1.6 See section 4.2.6 of this document concerning hollerith data types.

6.1 Some systems do not allow mixed mode arithmetic and the results on those that do may not be the same. Programmers should avoid mixing variable types in an expression except for those cases specified in the ANSI standard in lines 33-48 of this section. The permitted exceptions are:

a. exponentiation of any type by an integer primary.

b. exponentiation of real or double precision by either a real or a double precision primary.

c. real elements mixed in either double precision or complex expressions.

6.2 Relational expressions should be used only in logical IF statements and not in assignment statements. See also section 4.2.5 of this document.

6.3 Logical expressions should be used only in logical IF statements and not in assignment statements. Since some systems do not allow the use of parenthesis to force the order of evaluation of logical expressions, such expressions should be kept simple. See also section 4.2.5 of this document.

6.4 Take note of lines 3-8 in this section of the ANSI standard concerning evaluation of expressions containing functions. Side effects of the function on other items in the expression are not allowed.

7.1.1.2 Logical assignment statements should not be used. See also section 4.2.5 of this document.

7.1.1.3 The ASSIGN statement should not be used since some implementations do not have it.

7.1.2.1.2 The assigned GOTO should not be used since some implementations do not have it.

7.1.2.1.3 The value of a variable used in a computed GOTO should be checked and an error message printed when it is out of range. The action taken by various implementations in such cases is different and cannot be relied upon when a program is moved.

7.1.2.7.1 When a STOP statement is executed, some implementations accept an argument to the statement and print it out as in indication of which STOP has been executed. Since this is not used by all implementations it should be avoided.

7.1.2.7.2 The PAUSE statement should not be used since many large systems do not allow it.

7.1.2.8.1 If the initial parameter of a DO loop is greater than the terminal parameter the results cannot be guaranteed between different systems. This condition should be checked by the program before entering a DO loop.

7.1.2.8.2 Extended ranges of DO loops should not be used. Also note that ANSI specifications do not allow the control variable or loop parameters to be altered within a DO loop.

7.1.3 Any unit numbers in READ or WRITE statements should be specified as variables whose values are assigned and commented at the beginning of the program.

7.1.3.4 The character "+" as a carriage control in a FORMAT statement causes "no advance" according to ANSI standard. No advance is taken as meaning that the record overprints the previous record; in other words the print is preceded by a carriage return but no line feed.

7.2.1.1.1 Any use which depends on the internal representation of arrays should be avoided. Implementation differences can cause problems if COMMON or EQUIVALENCE is used with arrays of different sizes or different numbers of dimensions.

7.2.1.1.2 Adjustable dimensions in subroutines should be avoided since some systems do not allow them.

7.2.1.3.1.1 Since not all machines use the same amount of storage for various data types, programming should be independent of such internal representations. COMMON statements in different program units should contain identical data types, for example, and not use a real variable to take the space in COMMON used elsewhere by an integer variable.

7.2.1.4 EQUIVALENCE should be avoided in the interest of keeping a program readable and its effects clear.

7.2.1.6 See section 4.2.5 regarding logical variables.

7.2.2 Some implementations will repeat a DATA field until an array is filled. This convention should not be used. There should be a one to one correspondence between variable elements and data elements.

7.2.3.1 (1) It was agreed not to use G, D, or L conversion codes in FORMAT statements because some implementations do not have them. The restriction on D may be hard to stick to. (2) Scale factors should not be used since some implementations do not have them.

7.2.3.5 Scale factors should not be used.

Table 3 and 4. Some systems do not have all the functions in double precision and complex. If these must be used there should be a list in comment lines at the top of the program of those which are used.

8.3 Random number functions are used at all the CONDUIT centers for various modeling and simulation applications. These routines are typically machine dependent and cause much difficulty since test runs cannot be verified when the program is implemented at another center. To solve this problem each center has a test deck of random numbers and a substitute subroutine to pick the next number from the deck (or a file containing it). All sample runs should be made with the table lookup routine in the place of the random number generator. A listing and deck of the random numbers used should be included with the program when it is sent to another center. When the program has been implemented, and the results verified using a table lookup routine, the random number generator available at that center can be substituted.

It is assumed that the random numbers are evenly distributed between 0 and 1.

9.1.2 Some systems require a PROGRAM statement at the beginning of a program body. This statement may also set up files for the program. This is not ANSI standard and should have comments in the program describing what files are being set up for what purposes and the logical unit numbers that they will have.

10.2.6 In some implementations DATA statements in a subroutine renew their effect every time the subroutine is called, while other variables in the subroutine cannot be guaranteed across calls to that subroutine. In other implementations the variables retain their values across calls to the subroutine. Since differences in overlay techniques affect the way this works on different systems, variables should be initialized at the beginning of subroutines and not be expected to retain their values between calls. Variables which appear in DATA statements in subroutines should not be altered in those subroutines.

11. Items not mentioned in ANSI standard or in this document should be avoided. For example, ENCODE and DECODE, BUFFERIN and BUFFEROUT, LIBRARY, or CHAIN. Where they cannot be avoided, they should be commented within the program so that the programmer receiving it will know what is intended and be able to replace the code with something that will work at his center.

APPENDIX G

Working Paper #4

Editorial Aids for Importing and Exporting Fortran Programs on DTSS

CONDUIT

Editorial Aids for Importing and Exporting Fortran Programs on DTSS

by

Christopher G. Hoogendyk

A consortium of regional networks at
Oregon State University, North Carolina Educational Computing Service,
Dartmouth College, and the Universities of Iowa and Texas (Austin).

Editorial Aids for Importing and Exporting Fortran Programs on DTSS

by

Christopher G. Hoogendyk

October 1973

Dartmouth-CONDUIT
Working Paper #4

DRAFT COPY -- not to be reproduced without the written permission of the author.

Editorial Aids for Importing and Exporting Fortran Programs on DTSS

Introduction

The average academic computer center is batch processing, card oriented, and uses mostly Fortran. Dartmouth is time-sharing, terminal oriented, and uses mostly Basic. Because of these differences, there are problems which invariably arise in exchanging computer materials with other centers and which can be alleviated to some degree through the use of text processing computer programs. To this end, a small collection of such programs have been developed by Project CONDUIT. These programs are written in Dartmouth Basic 6th Edition and are in the CONDUIT sublibrary CONLIB***. They are:

- FREECARD
- LINELEN
- SEQUENCE
- CARDFREE
- DESEQ
- FORMATS
- FRTAN
- COMMENTS

Each of these programs will be described in detail below.

Exporting Fortran programs

On a card oriented system, Fortran programs are written in what will be referred to in this paper as card image format. That is, rather than having lines of the program free format as on DTSS, there is a strict adherence to the columns on the card. Columns 1-5 contain the statement number, column 6 contains the continuation character, columns 7-72 contain the Fortran statement, and columns 73-80 are reserved for sequencing numbers. If a "C" is punched in column 1 then that card (line) is a comment. A card image file does not have line numbers in front of each line.

FREECARD: This program converts a free format DTSS Fortran program (* for comment, and & for continuation) to a card image Fortran program. No changes are made in program content; it is simply reformatted to adhere to the correct column numbers. FREECARD works on the user's current file and the output becomes the user's new current file after the execution is successfully completed. To use FREECARD, type:

```
OLD    program-name
```

Where program-name is the name of the file to be edited . When the computer responds READY, type:

```
EXECUTE CONLIB***:FREECARD
```

When the computer again responds READY, the resulting file may be LISTed, REPLACEd, RENAMEd, and SAVEd, or whatever is desired. It should be a standard precaution after using any editing program to spot check the results before typing REPLACE in order to prevent any needless loss of programs.

LINELEN: When sending any file or program to another system, the length of the lines should be checked. A line that is too long can cause a record length error when putting the file on tape or cards. Also, most Fortran compilers on card oriented systems do not look past column 72, so anything that is beyond that column will be lost and may result in a faulty program. LINELEN is a program which checks line lengths in the user's current file. To use it, type:

```
EXECUTE CONLIB***:LINELEN
```

The program will ask for a maximum line length. Normally a maximum length of 72 should be entered. The user's current file will then be searched, and any lines longer than the specified maximum will be printed out. The user's current file will not be altered. Any offending lines in the program may then be altered by the user.

SEQUENCE: Some card oriented systems prefer to receive programs with sequencing numbers in columns 73-80. These numbers make it easier to spot check and assure that the program was received intact. SEQUENCE works on the user's current file. It checks line lengths against a maximum of 72 and adds sequencing numbers in columns 73-80. To use SEQUENCE, type:

```
EXECUTE CONLIB***:SEQUENCE
```

When the editing is complete, the resulting file will become the user's new current file and may be LISTed, REPLACEd, or whatever.

Importing Fortran Programs

There are two ways to get an imported Fortran program running on DTSS. First, since Dartmouth is developing a good ANSI standard Fortran, running the imported Fortran program in Fortran is becoming a viable alternative. The second alternative, and that which has been used in the past, is to translate the Fortran program into Basic. There are programs available to aid the user in both alternatives.

CARDFREE: This program reformats a card image Fortran program into a free format, line numbered Fortran program suitable for use with DTSS Fortran. It works on the user's current file, and fater successful completion the output file becomes the user's new current file. To use CARDFREE, type:

```
EXECUTE CONLIB***:CARDFREE
```

This will only reformat the program and will not change any of the content. To find out what programming changes have to be made to get the program running, the user will have to rely on program documentation and error messages from DTSS Fortran. Working Paper #3, "Agreeable Fortran", contains notes on language differences between the five CONDUIT centers and may be helpful.

DESEQ: This program will remove sequencing numbers in columns 73-80 and trailing spaces from a card image Fortran program. This usually saves 50% or more of the file storage required to keep such a program. DESEQ works on the user's current file, and upon successful completion the output becomes the user's new current file. To use, type:

```
EXECUTE CONLIB***:DESEQ
```

When the computer responds READY, the result may be LISTed, REPLACEd, or whatever.

FORMATS: One annoying characteristic of Fortran is that the FORMAT used in conjunction with a PRINT or WRITE isn't always in the same place in the program. Depending on the style of the person who wrote the program, the FORMAT statements could be either grouped or scattered anywhere in the program. FORMATS will search through a card image Fortran program, locate all the FORMAT statements, sort them according to the statement numbers, and group them immediately before the END statement of the program. If there is more statements for each one will be placed before the END statement of that particular program unit. With the FORMAT statements sorted and grouped in this manner, a particular one can be located without any trouble. To use FORMAT, the card image Fortran program to be edited should be the user's current file. Then type:

```
EXECUTE CONLIB***:FORMATS
```

Upon successful completion the results of the editing will become the user's new current file and may be LISTed, REPLACEd, or whatever.

FTRAN: If a decision is made to run an imported Fortran program in Basic, this program makes the task much easier. FTRAN takes a card image Fortran program and produces an output file in which most of the program has been translated into Dartmouth Basic. To use FTRAN, type:

```
.OLD CONLIB***:FTRAN.RUN
```

It will then ask for the names of an input file and an output file. The names of the files should be typed on the same line separated by a comma. If one of the files has a password then the name and password can be entered using quotation marks:

```
"Filename,password", Filename 2
```

where the appropriate file names and passwords are used. If the run exceeds the user's run time limit, FTRAN can be run in Background by writing a Background program as follows:

```
100 RUN
110 OLD CONLIB***:FTRAN
120 LINPUT filename,filename
130 OUTPUT errorfile
140 END
```

where the two filenames in line 120 are the names of the input and output files, and "errorfile" in line 130 is the name of any empty file in the user's catalog into which the Background run and any error messages will be written. To submit this Background job, SAVE it and type:

```
BACK
```

If the Background program is correctly written the computer will respond "***JOB ACCEPTED". The user can then do something else until the job is completed. See TM053, "Introduction to Background", for any questions on the use of Background.

There are five steps that should be taken in using FTRAN that will make the job considerably easier.

1. Once a clean copy of the card image Fortran program is available, some preliminary editing may be needed. The programs DESEQ and FORMATS can be used for this purpose. The program should also be checked to make sure that the correct character code was used. For example, open and close parenthesis may appear as percent and dollar signs if the wrong character set is specified for reading in a card deck.

2. Run FTRAN giving it the input and output file to work with.

3. Get a clean listing of the input and output files. To make the fourth step easier, the card image Fortran program used as input should have line numbers put on it starting at 1000 and going in increments of 100.

To do this, call up the program and type:

```
EDIT SEQUENCE 1000,100
```

When the computer responds READY, REPLACE the program. Also, the Basic program in the output file needs to be sorted. To do this, call up the program and type:

```
SORT
```

When the computer responds READY, the program can be REPLACEd. When this has been done, a Background program can be written to PRINT the input and output files on the printer.

4. Now set the two printouts side by side and correct all lines that have been commented by FTRAN. The line numbers on the Fortran program should correspond exactly with the line numbers in the Basic program if step 3 has been followed correctly. Problems that will be encountered and how to handle them will be discussed in detail below.

5. When all the corrections have been made, RUN the Basic program. Usually there will be some additional corrections to be made, and sometimes there will be some rather difficult problems encountered. See the discussion of these problems below. When the program seems to run successfully, there should be some test runs with the program that can be used to verify the correctness of the translated version. Try out the test runs and compare the results with those obtained by the original computing center. Always keep the line numbered listing of the Fortran program handy so that the Basic translation can be checked, and <u>do not</u> EDIT RESEQUENCE the Basic program until the results have been verified and the translation is considered final. At that time the table of variable names can be EDIT DELETED, a header of comments can be added to describe the origin and purpose of the program and who translated it, and the program can be EDIT RESEQUENCEd.

There are several problems that may be encountered in steps 4 and 5. Lines that were not translated and which may cause problems get flagged with a comment by FTRAN. These error comments fall into three categories:

1. WHAT??
2. CHECK THIS
3. UNDEFINED ARRAY OR FUNCTION NAME

Here is an example of what a section of program might look like in the Fortran version and in the Basic version:

```
FORTRAN                          BASIC

1000 IF (X) 350,350,371          1000 IF X1 <= 0 THEN 2100
                                 1001 GOTO 2300
1100 216 Y=X
                                 1100 LET Y1=X1
1200 GO TO 371
       .                         1200 GO TO 2300
       .                                  .
       .                                  .
       .                                  .
                                          .
2100 350 CONTINUE                2100 NEXT I4

2200 ASSIGN 350 TO Z             2200 ASSIGN 350 TO Z 'WHAT???

2300 371 CONTINUE                2300 NEXT J4

2400 PRINT 100,X,Y               2400 PRINT 100,X1,Y1   CHECK THIS
```

There are a couple of things to notice in this example.

1. There is a direct correspondence between line numbers. Line 1000 took two lines in Basic, but they were numbered 1000 and 1001 so that line 1100 in the Fortran program still becomes 1100 in the Basic program.

2. FTRAN added WHAT??? onto line 2200. This means that it didn't translate line 2200 at all but copied it directly from the Fortran program. This line will have to be translated by hand.

3. Line 2400 has the comment CHECK THIS. Lines flagged with this comment have had the variable names translated. The line will have to be retyped to incorporate FORMAT statement 100, but the variables do not need to be looked up in the variable table. The lines with "WHAT???" added to them do need the variables translated, and for this purpose a table of all the variable translations is added to the beginning of the Basic program.

The error message "UNDEFINED ARRAY OR FUNCTION NAME" means that a name was encountered followed by an open parenthesis and the name was not declared and was not a standard supplied function. These errors might result from program fragmentation when it was imported, non-standard supplied functions, or a user defined function. It should be noted that in Fortran all lists and arrays must be declared. If a program statement appears to be assigning some value to an element of an undefined list, that statement is probably a statement function. It should be translated into Basic as follows:

```
FORTRAN:   NAME (X,Y) = SIN(X)*Y**2

BASIC:     DEF FNN(X,Y) = SIN(X)*Y↑2
```

Then every occurrence of "NAME(" in the program should be replaced by "FNN(". If this replacement is done with the text editor (as it should be), the program will be left with error comments on lines that have already been corrected. See the description of the program COMMENTS below for an easy way to get rid of specific comments.

Some supplied functions are standard but do not have any corresponding function in Basic. These can be translated using function definitions. A couple of examples that may cuase problems are:

```
FORTRAN                    BASIC

SIGN  (X,Y)                100 DEF FNS  (X,Y)
                           110 LET FNS=ABS(X)
                           120 IF Y < = 0 THEN 140
                           130 LET FNS = -FNS
                           140 FNEND

LOG10(X)                   100 DEF FNL(X) = LOG(X)/LOG(10)
```

The first example is a source of problems because Basic defines the sign of a number (SGN) as having a value of -1,0, or +1, whereas in Fortran the 0 is grouped with the +1. The second example requires a little thought and is not terribly accurate. However, it is accurate enough for most uses.

There are a few other specific things that can cause problems. DATA statements are implemented differently at different centers. In DTSS Fortran the number of items of data must be the same as the number of variables. That is, if a list is dimensioned at 25, it must have 25 elements of data associated with it in a DATA statement. In some Fortrans the data field is repeated just like a FORMAT statement. If a program contains two lines

```
REAL NAME  (2,25)
DATA NAME  /1,2/
```

then the two item field is repeated until the entire array is filled.

String or character variables and caracter codes are different everywhere. If an imported Fortran program uses these types of variables a lot, then that program should be documented well enough to describe exactly what is being done. Code containing these variables has to be translated by hand.

The one problem which causes the most trouble is the use of COMMON and EQUIVALENCE with subroutines and function subroutines. These are flagged by FTRAN and need careful attention. Variables in COMMON should be added to the variable lists on the Basic SUB and CALL statements in the same order as they appear in COMMON. Labeled COMMON should be separated by name and each block treated independently. The following example demonstrates this:

```
FORTRAN                          BASIC

COMMON  /A/V1,  V2

COMMON  /B/V3,  V4,  V5

CALL SUB1(I,J,K,)                CALL "SUB1":I,J,K,V1,V2,V3,V4,V5

END                              END

SUBROUTINE SUB1(L,M,N)           SUB "SUB1":L,M,N,V4,V5,V1,V2,V3

COMMON  /B/V1,V2,V3

COMMON   /A/V4,V5

END                              SUBEND
```

Special care should be taken in dealing with lists and arrays in COMMON when they are not dimensioned the same. ANSI standard and most implementation of Fortrans specify that subscripted variables are stored in memory in such an order that the first subscript varies most rapidly. Using this as a guide, messy COMMONS can be unravelled.

Equivalences usually involve simple substitution of variable names with the text editor. However, there is sometimes a reason for using them such as character and integer variables to do manipulations of character codes. Such uses should be documented within the program.

The most difficult case to deal with is COMMON and EQUIVALENCE in a subroutine function. Such a function presents problems because in Fortran the variables not in the argument list are local to the function, whereas in Basic the function has to be moved into the main program and the variables can affect variables of the same name in the main program. If the function is passed to a subprogram, variables not on the argument list will point back to the variables in the main program. Language differences in a messy situation like this can cause incredible problems. If too much difficulty is encountered getting correct results the easiest solution is to change the function to a Basic subprogram and alter the function calls:

```
FORTRAN                          BASIC

COMMON /A/V1,V2

RSLT=FUNC(X9,ALT)*COS(RT1)       CALL "FUNC":X9,ALT,V1,V2,Q9
                                 LET RSLT=Q9*COS(RT1)

END                              END

FUNCTION FUNC(X,Y)               SUB "FUNC":X,Y,A5,A6,Q9

COMMON /A/A5,A6

FUNC = A5*A6+X**COS(Y)           LET Q9 = A5*A6+X↑COS(Y)

END                              SUBEND
```

Of course, all the variable names would be changed in the Basic version. The important point is that the COMMON variables are added to the call list and can be passed appropriate values or variables from subprograms. Also, variables which were local in the Fortran version remain local in the Basic version, and an extra variable has been added to take the place of the function value. Any expressions referring to the function have to be broken into several lines to accomodate the call statements. If the function is passed to a subprogram, then a variable can be passed that indicates the correct subprogram to call and an ON GO TO can be used. This can get messy, but it is a hazard of using complex language features. Such features should be used with discretion when it is expected that a program may be exchanged with other computer centers.

COMMENTS; After correction the output file from FTRAN using the text editor, a number of flag comments may still be on the program. COMMENTS provides a quick and easy means of removing them. It works on the user's current file and removes comments from specified lines. The output becomes the user's new current file after a successful run. To use it, call up the file to be edited and type:

```
EXECUTE CONLIB***:COMMENTS
```

A question mark will be printed on the terminal asking for input. The user should type in a list of line numbers in ascending order. COMMENTS will then print another question mark and the user may enter more line numbers continuing in ascending order from the end of the last list. When there are no more lines to be specified, a simple carriage return in response to the question mark will terminate the program. The user's new current file will have the comments removed from the specified lines. It is usually safer to use the text editor and COMMENTS to clean up large groups of arithmetic formulas, because retyping often results in typographical errors which can be difficult to locate when the program does not test properly.

APPENDIX H

Working Paper #5

The Export System

CONDUIT

The Export System

John M. Nevison

A consortium of regional networks at
Oregon State University, North Carolina Educational Computing Service,
Dartmouth College, and the Universities of Iowa and Texas (Austin).

The Export System

John M. Nevison

May 1974

Dartmouth-CONDUIT
Working Paper #5

```
*******************************************************************
* DRAFT COPY - Not to be reproduced without the written consent  *
* of the authcr.                                                  *
*******************************************************************
```

The Export System

Until recently each request for a set of programs from the computation center has been a special event. Special events cost people lots of wasted time and effort.

The Export system is a series of programs designed to store export versions of programs on magnetic tape and recover them in a form suitable for mailing to a customer.

Storing the programs is a job of the program librarian. Thirty programs can be done in about an hour.

Recovering the programs is a job of the corresponding secretary. He can recover a set and do the accompanying paperwork in under ten minutes. The response should be in the mail to the customer within 24 hours.

Storing a Program

Prepared programs and Q-files are approved by the Manager of Dartmouth-CONDUIT and forwarded to the Program Librarian who:

1. Writes all files on the storage tape.
2. Registers on-line for each file:
    a. Position on the storage tape
    b. Name
    c. Length in words
    d. Length in blocks
    e. Length in lines
3. Runs REPOSIT and addresses the set by subject, author, program name, and for each program, records
    a. Name
    b. Name of storage tape
    c. Position on storage tape
    d. L for list or LT for list and tape output
    e. Length in characters
    f. Length in lines
4. Informs the COMBIB editor of a new set for sale.

Exporting a Program

When a corresponding secretary receives an order for a program set, he (or she):

1. Makes sure payment is enclosed (or returns letter with a note requesting pre-payment).
2. Runs EXPORT for the appropriate sets.
3. Starts EXP-BACK and buys a new magnetic tape from the operator for the program to write on.
4. Writes a cover letter, prepares a tape sheet.
5. Mails
    a. Cover letter
    b. Tape sheet
    c. Magnetic tape
    d. Listing

People to Contact

See the Manager, the Chief Programmer or the Program Librarian for a detailed run through of the system.

APPENDIX I

Working Paper #6

How COMBIB Works

CONDUIT

How COMBIB Works

John M. Nevison

A consortium of regional networks at
Oregon State University, North Carolina Educational Computing Service,
Dartmouth College, and the Universities of Iowa and Texas (Austin).

How COMBIB Works

John M. Nevison

May 1974

Dartmouth-CONDUIT

Working Paper #6

```
**********************************************************************
* DRAFT COPY - Not to be reproduced without the written permission *
* of the author.                                                     *
**********************************************************************
```

COMBIB was designed to answer the teacher's question, "What does CONDUIT have for me?" If the interrogation is at a teletype terminal, the answer will take no more than ten minutes and will be in writing that he may take with him. The answer will be taylored by the teacher to his own interests.

A system that so serves the teacher can also provide the service organization itself with an up-to-date record of the status of program sets associated with books and booklets in its bibliography.

The bibliography arranges references by subject. To each reference are appended details on its recent history, "Commentary"; who has various versions of the computer programs, "Languages; and details about the programs, "Programs."

A reference in COMBIB looks like the illustration in the Appendix. It is a book or booklet describing the use of a computing idea in regular courses (not in Computer Science) in college or graduate school. It contains:

1. The author, and reference
2. The cost and how to order
3. Who recommended it
4. A list of appropriate details.

Additions and Corrections

All changes to COMBIB are made by the editor. He (or she) adds new entries, edits existing entries, and deletes old entries. He obtains his information from a variety of sources, but checks each one with a qualified reviewer before entering it in COMBIB. As a further control, he returns a sample of the listing to the individual who recommended the book and to the book's author and publisher.

Every paragraph in the Commentary on the book bears the initials of its author and date of entry. The first paragraph can be used as the short description included in an annotated bibliography.

Major alterations of a reference are sent to the author and to other concerned individuals.

In June of 1975 there will be a complete house cleaning of COMBIB. This will be repeated every second year.

COMBIB Services

In addition to answering the immediate questions of a teacher at a teletype, COMBIB may be used in other fashions. The editor can release a catalog of the complete contents of COMBIB. Such a catalog is not available to the general public at this time.

The editor can also release an annotated bibliography of the entire contents of COMBIB or of a selected discipline. These are available from the editor for a fee. COMBIB will also collect statistics on its use which the editor can monitor.

To Try COMBIB

If you wish to try COMBIB from your terminal (teletype):

1. Call 603-646-5171 10 char/sec
   -5141 15-30 char/sec
2. Type the user number R13999 and push the carriage return key
3. Type the password CON
4. Ask for an OLD program
5. Type the old program name (sixteen characters long)

   CONLIB***:COMBIB

6. When the machine is ready, type RUN and carriage return.

```
PLEASE PUSH RETURN, LINE 314
PP!!!!!!P!P!P!!F3^^3^03^^FX^OD=S^Z*

DARTMOUTH TIME-SHARING
LINE 314 ON AT 09:24  06 JUN 74, 096 USERS
LIST UCNEWS*** 05/29/74.
USERS' GROUP MEETING TODAY, 3:30, 102 KIEWIT. EVERYONE WELCOME.

USER NUMBER--P13399
COM HHHHHHH <-- PASSWORD
NEW OR OLD--OLD COMLIB***:COMBIB
READY

RUN

COMBIB     (COMPILED)  06 JUN 74  09:25


         ON LINE BIBLIOGRAPHY

 DO YOU WANT INSTRUCTIONS? NO

 SUBJECT #? LIST

  1 BIOLOGY
  2 BUSINESS
  3 CHEMISTRY
  4 ECONOMICS
  5 GEOGRAPHY
  6 MATHEMATICS
  7 MISCELLANEOUS
  8 PHYSICS
  9 POLITICAL SCIENCE
  10 SOCIOLOGY

 SUBJECT #? 6

          MATHEMATICS


 ITEM #? LIST

  1 CONDUIT DISCIPLINE COMMITTEE
  2 Basic Programming
  3 Calc and Cmptr
  4 Calc and the Cmptr Revolution
  5 Calc of Population
  6 Calc with Cmptr Applctns
  7 Calc:  Cmptr Oriented Prsntatn
  8 Comp Applications for Calc
  9 Comp-oriented Calc Probs
  10 Diff Eq's
  11 Elem Num Anal
  12 Ftran Digicomp Programmng
  13 Intro to Comp Mthds for Calc Stdnts
  14 Lin Alg
  15 Num Computing
  16 Single Variable Calculus
```

```
ITEM #? 14

McLaughlin, Donald E.  "A Computer Oriented Course in Linear Algebra."
     University of Iowa, 1973.
     Cost:  $2.00 from Regional Computer Center, University of Iowa,
            Iowa City, Iowa   52242
     Recommended by: A. Ziebur

DETAILS AVAILABLE:

 1 Commentary
 2 Programming Languages


DETAIL #? 1

          Commentary

     This text was the subject of a Mathematics Conduit national
workshop at the University of Iowa in July of 1972.     JMN

     This book has been used by Darrien Moore at the University of
New Hampshire in Durham, N.H.  Anyone in the Dartmouth region
having questions about his experience should contact him at the
Department of Mathematics.  JMN


ANOTHER DETAIL #? 2

          Programming Languages

FORTRAN -- Ted Sjoerdsma
           Computer Center
           University of Iowa
           Iowa City, Iowa 52240
BASIC -- John M. Nevison
         Kiewit Computation Center
         Dartmouth College
         Hanover, NH    03755


ANOTHER DETAIL #? NO

ANOTHER ITEM #? NO

ANOTHER SUBJECT #? NO


THANK YOU, CALL AGAIN.


0.918 SEC.  136 I/O
READY
```

```
SUBLIS    06 JUN 74  09:49

ENTER MAXIMUM MARGIN? 72
COMPLETE OR ANNOTATED LISTING? ANNOTATED
DO YOU WANT ONLY CONDUIT LISTINGS? NO
DO YOU WANT ALL SUBJECTS? NO
INPUT THE SUBJECT NUMBER THAT YOU WANT? 6


                    ******************************
                              MATHEMATICS
                    ******************************
```

(Exerpt of full Contents)

```
-----------------------------------------

Kemeny, John G. & Kurtz, Thomas E.  "Basic Programming"  2nd ed.
    New York: John Wiley and Sons, 1971.
    Cost -- $5.95/copy   (paper)
    *Recommended by:  Committee

        Part I is an elementary introduction to computer programming in
    BASIC.  Part 2 gives concise explanations of about 35 different
    applications, mathematical and otherwise, in areas ranging from
    business to music, as well as explaining the more advance features of
    BASIC.  Many exercises, and each chapter in Part 2 concludes with
    suggestions for projects.
        DAS

-----------------------------------------

Smith, David A.  "Interface:  Calculus and the Computer"
    (Preliminary version) Sample copies available from the author;
    copies for classroom use available for cost of production and
    mailing.  Test to be published by Houghton-Mifflin in early
    1975.
    Recommended by: David A. Smith

        Supplementary text intended for use with any standard calculus
   book; elementary introduction to computers and programming, using the
    CPS dialect of PL/1; heuristic treatment forms the mathematical basis
    for the rest of the book; covers solution of equations, max/min
    problems, numerical integrations and differentiation, polynomial
    approximations to transcendental functions, initial value problems;
    includes "lab manual" sections on each topic.   D.A.S.

-----------------------------------------

Hamming, Richard W.  "Calculus and the Computer Revolution"
    Boston:  Houghton-Mifflin Co., 1968.
    Cost -- $1.75/copy
    *Recommended by:  David A. Smith


        Supplementary text for a calculus course; brief chapters on
    roundoff problems, use of finite processes to approximate infinite
    ones; numerical solution of differential equations, algorithms, and
    general symbol manipulation.  Exercises after each chapter, except the
    last.  No programming instruction or exercises.   D.A.S.
```

Bogart, Kenneth P. & Michael P. Vitale. "The Calculus of Population"
Project Compute, Dartmouth College, Hanover, N.H., 1973, ??pp.
Cost: $4.00 from Project Compute, Kiewit Computation Center,
Dartmouth College, Hanover, N.H. 03755
*Recommended by: Allen Ziebur

An informal, applications-oriented introduction to the theory of calculus. The computer's role in helping students to understand concepts and find solutions to problems is emphasized. The text is designed for supplementary use in an elementary functions course, or as an introduction to a more rigorous course in calculus. It assumes a background of high school algebra (laws of exponents, rational numbers, graphing, absolute value, sets) and exposure to elementary programming in BASIC or related languages. A.K.M.

Lynch, Ransom V. & Ostberg, Donald R. & Kuller, Robert G. "Calculus with Computer Applications" Lexington, MA: Xerox College Publishing, 1973.
Cost: ???
*Recommended by: David A. Smith

Complete text for three-semester calculus course; conventional topics and approach, plus attention to numerical methods where appropriate, plus 77-page appendix on "Applications of Computing to Calculus." The appendix covers algorithms and flowcharts (no programming), differentiation, max/min problems, solution of equations, limits, integration, computation of transcendental functions, series, differential equations. D.A.S.

Walker, Robert J. & Stenberg, Warren, & others. "Calculus: A Computer Oriented Presentation." Florida State University, 1970.
Cost--??? from Center for Research in College Instruction in Science and Mathematics (CRICISAM), Florida State University,
Tallahassee, Florida 32306
*Recommended by: David A. Smith

Complete text for a one-year integrated calculus-with-computer sequence, with emphasis on algorithmic methods, and with an error-control approach to epsilons and deltas. Outgrowth of an NSF-funded project. D.A.S.

Dorn, William S. & Gary G. Bitter & David L. Hector, "Computer Applications for Calculus" Boston: Prindle, Weber, and Schmidt, Inc., 1972.
Cost -- ???
*Recommended by: David A. Smith

Supplementary text intended for use with any standard calculus book; concentrates on use of the computer to illustrate ideas normally encountered in the standard course: functions, limits, derivatives, integrals, differential equations, sequences and series; also chapters on root finding, linear algebra, and miscellaneous topics. Ample supply of exercises; no programming instruction, but all sample programs are presented in both FORTRAN and BASIC, with sample results obtained in conversational mode. D.A.S.

APPENDIX J

COMPUTe Publications

ABSTRACTS and STATUS
of
TEXTS in PROCESS

Bulletin No. 4½

June 1974

Title: Survey Sampling in the Environmental Sciences: a Computer Approach

Authors: James P. Barrett & Mary E. Nutt
Department of Forestry, University of New Hampshire
Department of Zoology, University of New Hampshire

Abstract: Computer programs are used to approach the theory, planning, and application of sampling in the environmental sciences. In addition to five basic methods (simple random sampling, systematic sampling, stratified sampling, regression estimation, and cluster sampling), special techniques are presented for use in forestry and field zoology. The exercises cover a wide range of environmental issues and wherever practical the data are taken from current scientific research. The text is written for college undergraduates in the natural sciences and assumes little or no experience with statistics. The programs are in BASIC and require little or no programming ability.

In survey planning, the computer is used to investigate the interaction of precision and cost. For each sampling method, programs calculate the required number of samples and estimated cost at specified levels of precision and confidence. For each sampling method, programs compute means, totals, and confidence intervals as well as variances, standard errors and other details.

The book consists of ten chapters: Introduction, Populations and Samples, Sampling Theory, Simple Random Sampling, Systematic Sampling, Stratified Sampling, Regression Estimation, Cluster Sampling, Forestry, and Field Zoology. Programs are listed at the end of appropriate chapters and exercises are provided which rely heavily on the use of these programs.

Approximate length: 300 pages

16 programs written in BASIC are available with this package.

Status: First draft of text complete with the exception of Chapter 10. Anticipated completion date is early summer 1974.

Title: Cognitive Psychology: a Computer-Oriented Laboratory Manual

Author: William L. Bewley
Department of Psychology
Lawrence University

Abstract: This book provides a laboratory introduction to cognitive psychology. The material is presented by means of six experiments, each of which is run on a time-shared computer system using the BASIC language and a teletype terminal as the input/output device. In each experiment, the computer presents a task to the student and to a simulation of a model of human information processing relevant to the task. The student is asked to compare his performance with that of the model and to compare what he thinks he did in the task with what the model says he did. The simulations are designed so that features of the task and the model can be changed by the student, allowing him to run his own experiments on the model or on other humans. Questions are asked at the end of each experiment which lead the student to conduct such experiments.

The six experiments were chosen to cover the range of cognitive activity represented in the current general conception of the human information-processing system (e.g., Atkinson & Shiffrin, 1968; Norman and Rumelhart, 1970).

Experiment 1. Pattern Recognition
Task: visual search
Model: pandemonium (Selfridge, 1959)

Experiment 2. Short-Term Memory
Task: Continuous Memory
Model: buffer model (Atkinson & Shiffrin, 1968)

Experiment 3. Long-Term Memory
Task: Paired-associate learning
Model: Discrimination net (Hintzman, 1968)

Experiment 4. Concept Learning
Task: Blank trials task of Levine (1966)
Model: Several--sampling with replacement, local consistency, consistency check, focusing.

Experiment 5. Decision-Making
Task: Two-person game
Model: A combination of the social motives of Messick and McClintock (1968) and the linear operator model of Rapoport and Chammah (1965)

Experiment 6. Problem Solving
Task: Missionary and cannibals problem
Model: The General Problem Solver (Ernst & Newell, 1969).

The presentation of each experiment includes a brief abstract, a description of procedure, a description of the model, a set of study questions designed to encourage critical evaluation of the model and independent experimentation, and a general discussion which briefly describes alternative models and suggests further reading. In addition to the six experiments, the book contains an introduction describing the relation of each experiment to the general information-processing model, and two appendices, one dealing with a suggested format for lab reports and the other with the minimum chi square procedure for goodness of fit.

An instructor's manual contains program listings and documentation and discusses material not included in the student book which the instructor may wish to use as background for the experiments.

Length: Student manual--102 pages
Instructor's manual--96 pages

7 programs in BASIC are available with this package.

Status: Student manual--Final draft completed.
Instructor's manual--Final draft completed. AVAILABLE FOR CLASSROOM USE!

Title: The Calculus of Population (tentative)

Authors:
Kenneth P. Bogart & Michael R. Vitale
Mathematics Department, Dartmouth College
Mathematics Department, Skidmore College

Abstract: An informal, applications-oriented introduction to the theory of calculus. The computer's role in helping students to understand concepts and find solutions to problems is emphasized. The text is designed for supplementary use in an elementary functions course, or as an introduction to a more rigorous course in calculus. It assumes a background of high school algebra (laws of exponents, rational numbers, graphing, absolute value, sets) and exposure to elementary programming in BASIC or related languages.

Chapter 1 examines the growth characteristics of a population of paramecia, using this sample as a vehicle for developing methods of describing and predicting growth and rates of growth. Chapter 2 discusses the concept of a derivative in greater detail, demonstrating how to find the derivative of a variety of function types. In addition it discusses the concept of area and develops methods for finding the area of various regions of the plane. Chapter 3 considers the properties of the derivative in still more detail and establishes a connection between the derivative and the integral, demonstrating techniques for finding integrals. Chapter 4 synthesizes the content of the previous chapters by applying the results to the more complex situation of two species in a closed environment. Emphasis throughout is on mathematical representation of the properties of population, using this development as a rationale for the study of calculus. Exercises follow each chapter, many requiring the use of a computer. Programs in BASIC are included as part of the text.

Approximate length: 180 pages

36 programs written in BASIC are available with this package.

Status: Final draft complete.
AVAILABLE FOR CLASSROOM USE!

Title: An Introduction to Environmental Measurement and its Descriptive Analysis by Computer

Author: George F. Estabrook
Department of Botany
University of Michigan (Ann Arbor)

Abstract: Designed to serve as a specific guide to methodology in the investigation of environmental variation. The exposition is based on the contention that method is learned through its practice, and assumes active involvement on the part of the student in the study of some aspect(s) of environmental variation. Three specific data analysis techniques are discussed to exemplify the role of data analysis in scientific methodology. They serve the following purposes:

1) to quantify the similarity of pairs of objects in a study described with nominal or ordinal measurements, leading to discovery of the hierarchical classification involved;

2) to quantify the extent to which one nominal variable can be predicted from the knowledge of another, and to describe the nature of the correspondence which makes this prediction possible;

3) to represent spatially in a few dimensions the objects in a study described by interval or ratio variables, and to describe the relationship among the variables which makes such a representation possible.

The specific techniques were selected because they offer distinct approaches appropriate for different kinds of measurement analogies; because their methodology is sufficiently straightforward and simple to be accessible to students without sophisticated mathematical backgrounds; because they are general and offer wide potential applicability in environmental studies; and because they lend themselves to computer programs which can be used by students to exploit the techniques.

Significant use is made of the computer in analyzing study objects according to these techniques, and instruction is provided in the intelligent use of the programs written for the book. Various study-related tasks are also assigned the student as he progresses through the text.

Chapters: How to Use this Book; Measurement of the Environment; Representation of Data; Similarity Among Objects; Stochastic Dependence of Nominal Characters; Hierarchical Classification of Objects; Spatial Representation of Objects in a Study. Appendices include an extensive bibliography of pertinent literature and discussion of how to use the computer programs.

Approximate length: Text -- 85 pages in single-spaced draft; will be changed to double space in next version

Bibliography -- 10 pages
Other appendices -- 40 pages

Status: Final draft complete and being used by author in regular course.
AVAILABLE FOR CLASSROOM USE!

Three programs written in FORTRAN accompany the text.

tle: Physics of Energy and Environment

thor: Allan R. Evans
Department of Physics
University of California at Irvine
(currently: Societal Analysis Dept., GM Research Labs)

stract: An outgrowth of a course entitled "Physics of the Environment" taught by the author for two years, these computer-oriented materials have been developed to discuss the physics of environmental problems. Emphasis is on physical principles, model formulation, and quantitative solutions, with the computer used in an interactive mode for instruction and calculation. Essential introductory material studies the tools of quantitative reasoning, including concepts of rates of change and functional dependencies.

This material is intended for use in courses dealing expressly with environmental problems, or as supplementary material in traditional physics courses for non-majors. The following sections are included:

Part I. (No title yet)

A. Trends, Change, and the Scale of Problems
   *electrical energy consumption
   *simple and compound growth
   *compound growth of population
B. Growth and Consumption in the Environment
   *electrical energy consumption
   *U.S. population growth in present century
   *motor vehicle population
   *total energy consumption
   *electrical energy and total energy consumption
C. The Problem of Human Population
   *growth of population--characteristics
   *growth of population--models

Part II. The Energy Balance of the Earth

A. The Scale of Human Intervention
   *energy consumption
   *alteration of the earth's surface
   *air pollution
   *global temperature increase
   *climatic changes in cities
   *"fire and ice"

B. The Earth as a Blackbody
   *absorption and radiation of solar energy
   *physics of blackbody radiation
   *the atmospheric greenhouse
C. Climatological Models
   *the glass greenhouse
   *the atmospheric greenhouse
   *a model of glacier growth
   *more detailed climatological models
D. Evolution of Climate

roximate length: Student text -- 140 pages
Appendices -- 25-30 pages
Instructor's introduction -- ?

programs written in BASIC are included with this package.

tus: Text is in second draft. To be completed summer 1974.

Title: Models for Control of Man's Physical Environment

Authors:

Joseph J. Harrington &
Environmental Health
Engineering
Harvard University

Peter P. Rogers &
City Planning and
Env. Health Eng.
Center for Population Studies
Harvard Univ.

R. K. Bhuiya
Hydro-Quebec
Montreal
(formerly Center
Population Stu

Abstract: An outgrowth of a course by the same name taught by Harrington and Rogers for five years, the purpose of this material is to acquaint upper-level undergraduates with the use of mathematical and computer models currently used in environmental control. Models discussed are primarily digital computer simulations and mathematical programming, dealing with quality, treatment, and control of air and water, land use, solid waste disposal, and health epidemiology. These models are organized in the material for classroom and laboratory computer use and homework assignments.

18 programs in the BASIC language are available as part of the package.

Approximate length:
Student's text -- 75-100 pages
Instructor's introduction -- ?
Appendices -- ?

Status: Text is in second draft. Completion date uncertain.

Title: Investigations of Stack Gas Dispersion and Urban Air Pollution through Numerical Exercise

Author: Yuji Horie
Dept. of Environmental Sciences & Engineering
School of Public Health
University of North Carolina at Chapel Hill

Abstract: Several air pollution models are explained in a systematic fashion, with the objective of providing students with the best means for understanding the nature of air pollution problems. Charts and graphs from which answers can be found are minimized, and analytical methods and the use of computer programs are maximized. For each topic a simplified version of the problem is first explained by means of an analytical method. The more complex and realistic situations are then presented in conjunction with computer programs. Students thus have an option in solving a given problem and a means to check their results. An attempt is made to keep the mathematics at a level within the reach of most college undergraduates. Computer programs are fairly self-explanatory and should not cause any difficulty in their use.

Chapter 1 explains the meteorological fundamentals and the types of dispersion models for both reactive and non-reactive pollutants. The roles and practical usages of meteorological analysis are discussed in Chapter 2 together with the sources and availability of emission and meteorological data necessary for such analyses. Chapter 3 focuses on the problems associated with an elevated point source and studies the plume behavior and ground level concentrations both analytically and numerically. Chapter 4 describes the treatment of many sources and explains both the synoptic and climatological models of urban air pollution. In Chapter 5, selected topics such as environmental capacity and control strategies are presented.

Approximate length: Student's text--150 pages
Appendices--30 pages (programs)
Instructor's introduction--?

5 programs in the FORTRAN language are available with the package.

Status: Author's rough draft. Completion date uncertain.

Title: Physics Tutorial Problem Workbook

Author: John L. Jones
Dept. of Computer Science
United States Naval Academy

Abstract: The workbook consists of 50 tutorial problems and their related diagrams, designed to be worked by the student in conjunction with a computer operating in a time-sharing mode. The problems are designed to augment the classroom instruction for an undergraduate course in general physics. Half the problems deal with mechanics, the other half with electricity and magnetism. Each problem write-up includes background information, a list of problem objectives, and references to appropriate sections in Sears and Zemansky, Halliday and Resnick, and Shortley and Williams. The package includes 50 programs written in a common subset of the BASIC language, each of which leads the student through a given problem and gives him help if necessary.

The instructor's introduction includes a discussion of the rationale for using such material; suggestions for integrating it into a course; suggestions for making the programs operational on the local facility (whom to see, degree of program availability, etc.); and hints on ways to make the programs more sophisticated for specific purposes (e.g., use of a driver program with teacher-generated data files, provision for varying initial conditions during subsequent exposure to the same problem, provision for simple record-keeping), depending on features available on a given facility. Sample programs offering these features are appended to the teacher's material.

Approximate length: Student's text -- 100 pages
Instructor's introduction--37 pages

4 programs in BASIC are available with this package.

Status: Final draft complete; all programs written and operational.
AVAILABLE FOR CLASSROOM USE!

Title: A Computer Laboratory Manual for Number Theory

Author: Donald E. G. Malm
Department of Mathematics
Oakland University

Abstract: This laboratory manual is intended to supplement a text in number theory. It is arranged according to topic. For each topic, there is i) a description of the topic, its concepts, terms, theorems, etc.; ii) a set of experiments to work before studying the topic; iii) a set of experiments to work while studying the topic; iv) a set of more extensive, open-ended problems; v) references to texts and monographs. The instructor's manual contains detailed information on using the laboratory manual.

Chapters: Introduction, The Euclidean Algorithm and Factorization into Primes, Linear Diophantine Equations, Congruences and Multiplicative Functions, Magic Squares, Quadratic Residues, Continued Fractions, Diophantine Equations, Sums of Squares, Geometric Number Theory and Partitions.

Approximately 40 programs written in the BASIC language accompany the text.

Status: Being supported summer 1974. Completion date uncertain.

Title: Place: A Computer Manual in Introductory Geography

Author: Vincent H. Malmström
Department of Geography
Middlebury College

Abstract: In the first chapter, the author presents his definition of geography, together with a brief outline of the basic concepts and tools which the geographer employs, giving special attention to the potential of the computer. The second chapter deals with computer mapping, giving a step-by-step description of the procedure involved in collecting, coding, and printing out data with a variety of options. In Chapter 3 various aspects of map analysis by computer are explored, including the determination of scale and the measurement of area, distance, direction, and centrality. The statistical analysis of data is also investigated, as is the problem of defining regions by computer. Chapter 4 provides a series of examples of how the computer may be applied in physical geography, including locational determination (latitude and longitude), the analysis of weather and climate data, and evaluation of land use potential on the basis of a physical inventory. In the final chapter, some computer applications within the general area of cultural geography are examined, with illustrative problems being drawn from economic, historical, and political geography.

29 programs and 35 data files written in BASIC are available as part of this package.

Length: 167 pages

Status: Complete; AVAILABLE FOR CLASSROOM USE!

Title: Models in Ecology

Author: Jyri E. Paloheimo
Dept. of Zoology
University of Toronto

Abstract: A handbook principally designed to be part of a course in resource management, this material is the product of the author's five years' experience in teaching an undergraduate course titled "Mathematical Ecology". The material is aimed at students whose background is primarily in zoology, botany, or biology, rather than in mathematics. The author eases exposure to more sophisticated mathematical treatments by making use of the computer as a teaching aid (e.g., to generate visual representations of populations and their characteristics, to examine the behavior of mathematical models, to explore the patterns and relationships among data, etc.). Numerous problems are included in the text, some involving the use of a computer.

Students are led by the discussion to the point where they are capable of creating a program which models a single-species population living in a fixed resource environment and subject to a single form of predation. Subsequently, two such models are permitted to interact in order to study the effects of competition and the ways in which two-species systems differ from single-species systems. Eventually, students are shown how to approach the development of a model for cycling carbon, water, and perhaps other basic nutrients in a biosphere. The ultimate product is a simple ecosystem synthesized from the earlier experiences with population models and nutrient cycles.

The text includes the following chapters:

Remarks for teachers
Introduction: Mathematics and computers

I. Growth and Regulation
   *demography of man and animals
   *regulation of numbers and sizes

II. Spatial Aspects of Populations

III. Theory of Search: on Prey and Predator

IV. Management of a Resource
    *on whales and whaling
    *are Atlantic Salmon to survive?

V. Biology of Populations
    *a hypothetical population
    *on ecological and other indices of efficiency
    *computer paramecia
    *computer Daphnia

VI. Multispecies Systems
    *coexistence and competition
    *predatory food chain

VII. Ecosystem Models
    *a logistic model and beyond
    *on resource and nutrient circulation
    *a computer ecosystem

Appendices (programs with comments)

Approximate length: Student's text -- 150 pages
Instructor's introduction -- ?
Appendices -- 20 pages

All programs are written in BASIC.

Status: First half is in second draft; second half is in first draft. Completion date uncertain.

Title: ACRES: Area Community Real Estate Simulation

Author: George B. Pidot, Jr.
Department of Economics
Temple University

Abstract: ACRES is a gaming-simulation designed to give the players some insights into the dynamic interactions of a developing urban area. Players representing the business, residential, and government sectors must make a wide variety of decisions concerning the properties they may develop, thus affecting the well-being of the town in which they are located.

This game has been used in economics classes at Dartmouth and at other colleges using the Dartmouth Time-Sharing System.

The program is written in ANSI FORTRAN.

Approximate length: 120-140 pages

Status: Preliminary version AVAILABLE FOR CLASSROOM USE. A revised version, reflecting changes in the game, should be available in the near future.

Title: Optimal Location of Facilities

Author: Gerard Rushton
Department of Geography
University of Iowa

Abstract: This is an introductory text on methods for optimally locating facilities and on the preparation of data and control cards for computerized implementation of these methods for large problems. Oriented to the undergraduate student with no experience in computer use, the text is fully sufficient to lead the student to set up and solve these locational problems both by hand (when the problems are small) and on the computer (when the problems are large). It is intended for use in undergraduate classes in economic and social geography, urban and regional planning and spatial analysis.

Methods are described for locating single and multiple facilities to minimize the average distance separating facilities from the locationally dispersed and differentially weighted demand points; to minimize the maximum distance to the farthest demand point; to minimize the total distance subject to a maximum distance constraint; and to find the minimum number of facilities required for every demand point to be within a specified distance of a facility. In cases where it is more relevant to minimize time or cost than distance, it is shown how these methods can be adapted for this purpose. Distance minimization criteria are discussed with reference both to locations in continuous space and locations on a route structure. A chapter is devoted to the problem of comparing the shortest paths through a route structure by a specified combination of travel-route modes. Each problem is described in verbal and mathematical terms, methods of solution are described, detailed computations are given for a handworked example, and the input form is specified for computer programs which solve these problems. Specimen control cards are described for a variety of forms in which the problems are found. Homework problems are provided.

An introductory chapter discusses applications of these methods to problems in both public and private sectors of the economy and the special problems associated with their use in developing countries.

5 computer programs in FORTRAN are part of this package.

Approximate length: 160 pages

Status: First draft complete; should be available in mid-summer 1974.

Title: Descriptive Statistics

Author: Harrison D. Weed, Jr.
Mathematics Department
Dartmouth College

Abstract: Designed to be a supplement to standard textbooks on elementary statistics and probability. Provides a discussion of methods and their intelligent uses in descriptive statistics often missing in current texts, as well as a much needed introduction for the student to statistical calculations using the computer. Various methods for describing and summarizing raw data are discussed. Problem assignments are computer oriented.

Sections: Introduction, Classification of Data, Frequency Distributions, Measures of Location, Measures of Dispersion.

Approximate length: Student's text -- 84 pages
Instructor's introduction -- ?

Computer programs written in BASIC are available with this package.

Status: Preliminary version complete; should be available for classroom use by late summer 1974.

Summer 1974 COMPUTe Projects

* * *

Baird, John C. (Department of Psychology, Dartmouth College), and Elliot Noma (Programmer for the Social Sciences, Dartmouth College)

"Psychophysical Processes"

Dershem, Herbert (Department of Mathematics, Hope College)

"Exercise Manual for Computer Augmented Applied Statistics Course"

Lesk, Arthur (Department of Chemistry, Fairleigh Dickinson University)

"Computing Methods in Chemistry"

Malm, Donald E. G. (Department of Mathematics, Oakland University)

"A Computer Laboratory Manual for Number Theory"

Merriam, Daniel (Department of Geology, Syracuse University)

"Computer Fundamentals for Geologists"

Sommer, John (Department of Geography, Dartmouth College)

"Instructor's Text for Area Community Real Estate Simulation Game (ACRES)"

(And possibly)

Ciecka, James (DePaul University)

"Principles of Macro-Economics: A Computer-Assisted Approach"

APPENDIX K

CONLIB***:LIBCAT

CONLIB***:LIBCAT

Contents Of CONDUIT Program Library

Last updated May 29, 1974 by David Cooley.

Summary of program names:

Biology programs: EPOP, EXPOP, SIGPOP, RAND-K, COMPET-1, COMPET-2, COMPET-5, and TUNDRA.

Business and Economics programs: BALANCE, BEARSIM, BULLPUP, CRIT, DECIDE, EOQ, FIRM, FUTURE, INSYS, KEYNES, MACWH, MARK, QUESIM, SQC, and UNISIM.

Chemistry programs: 1DFOU, ABCANAL, ABC, BOX, CONTOUR, DECAY-1, DERTI, EDTA, ENTROPY, EQUIL, FRSTLW, GENPT, HA, HATOM-1, HDIAG, HMO, IDGAME, IONCH-1, IR, MASPEC-1, MATINV, MX, NERNST, NEWTON, NMR, NONLIN, POWAC, POWID, RADIAL-1, RADIO, SOL-1, STOIC, and VSEPR-1.

Mathematics programs: CHAREQ, ECHLN, GRAM, JACOBI, LINEAR, POLREG, kEDECH, and SIMPLEX.

Utility programs: CARDFREE, FREECARD, FORMATS, DESEQ, FTRAN, COMMENTS, LINELEN, SEQUENCE.

Programs with no suffix are working programs translated from FORTRAN to BASIC at Kiewit. Programs with a -X suffix have either not been thoroughly debugged or not been verified with test data and known results. Programs with a -1 suffix have been translated at Kiewit and modified by the people who are actually using them in classes.

To access a program in CONLIB***, type:

```
OLD CONLIB***:,program name.
```

where ,program name. is the name of the program as listed above, and then type "LIST" or "RUN" to get instructions.

If the program has not been saved, contact the CONDUIT OFFICE to have the program put into CONLIB***.

*********** BIOLOGY PROGRAMS ***********

"EPOP" -- An exponential growth model for unlimited growth of a single species in an optimal, infinite environment. Uses solution of a differential equation.

"EXPOP" -- This program is the same as "EPOP" except that it uses an incremental addition method (difference equations).

"SIGPOP" -- This program is the same as "EXPOP" with the addition of an environmental carrying capacity to demonstrate the sigmoid growth curve.

"RAND-K" -- This program is an addition to "SIGPOP" which provides a randomly varying carrying capacity.

"COMPET-1" -- A two species competition model based on the Lotka-Volterra equation and using incremental addition (difference equations).

"COMPET-2" -- The same as "COMPET-1" except with graphic output.

"COMPET-5" -- The same as "COMPET-1" with the addition of randomly varying carrying capacities and immigration or emigration based on population levels.

"TUNDRA" -- A simulation of the arctic tundra ecosystem. includes different species of grass, dead grass, lemmings, owls, and detritus. functions are separated into the three seasons, and real data are used to calculate the constants used in the program.

**** BUSINESS AND ECONOMICS PROGRAMS ****

"BALANCE" -- A heuristic program for solving assembly line balancing problems.

"BEARSIM" -- A monte carlo simulation program which demonstrates the effects of various maintenance policies.

"BULLPUP" -- A macroeconomic policy game.

"CRIT" -- Solves scheduling of activities within a project in terms of the critical path.

"DECIDE" -- Solves decision problems which are structured as a decision tree with both decision branches and chance event branches.

"EOQ" -- Computes the most economical inventory order quantity under a variety of conditions.

"FIRM" -- An economic simulation.

"FUTURE" -- Translates quantitative facts about the past into a forecast for the future.

"INSYS" -- Simulation to test effect of policy changes on inventory levels and factory output for a factory-wholesaler-retailer system.

"KEYNES" -- A macro-economic model.

"MACWH" -- Simulation based on the wharton econometric model of the U.S. Economy.

"MARK" -- Economics program.

"QUESIM" -- A simulation program for modeling a single phase multi-channell queueing system.

"SQC" -- Determines whether a process is in control or not based on the control charts for the mean and range of a group of sample data.

"UNISIM" -- Simulates the decisions faced by the operations manager in a one-product company.

********** CHEMISTRY PROGRAMS **********

"1DFOU" -- Executes a one dimensional fourier synthesis or structure determination for a centrosymmetric (two-atom) structure.

"ABC" -- Simulates the 15 line ABC nuclear magnetic resonence spectrum.

"ABCANAL" -- Finds the analytical solution to the system a<=>b<=>c involving 4 rate constants.

"BOX" -- Solves the one dimensional particle in a finite box problem.

"CONTOUR" -- Calculates and prints out electron density contour maps for various atomic, hybrid, and molecular orbitals.

"DECAY-1" -- Simulates either simultaneous or consecutive radioactive decay.

"DERTI" -- A generalized data reduction program for analysis by titration of up to 50 data pairs.

"EDTA" -- Simulates the m(2+)-edta titration.

"ENTROPY" -- Calculates the entropy of a molecule in the gas phase from spectroscopic data.

"EQUIL" -- Given the gas phase reaction: aa + bb <--> cc + dd EQUIL calculates the equilibrium constant by minimizing the total free energy function.

"FRSTLW" -- Program designed to aid students in the study of the first law of thermodynamics.

"HA" -- Simulates the weak acid - strong base titration.

"HATOM-1" -- Calculates electron transitions.

"HMO" -- Uses simple Huckel molecular orbital theory to approximate pi molecular energy levels, wave functions, electron densities, bond orders and free valencies.

"IDGAME" -- A qualitative analysis game in which student tries to identify an organic unknown spending the least amount of money for tests.

"IONCH-1" -- Calculates percent ionic character of a dipole and electronegativity of the two atoms.

"IR" -- Searches a database of organic compounds containing the name, melting and boiling point and the position of the strongest band in the IR spectrum.

"MASPEC-1" -- Simulation of the Ealing small mass spectrometer as described in Ealing Corporation's 1969 teaching catalog.

"NERNST" -- Simulates the potential of a m/m(z+) electrode in an aqueous ammonia solution as a function of ph.

"NEWTON" -- Uses the Newton-Rapheson method for systems of nonlinear equations.

"NMR" -- Simulates an ab, ab2, a2b3, a2x2, or abx nuclear magnetic resonance spectrum.

"NONLIN" -- Does a nonlinear least squares analysis.

"POWAC" -- Powder Arc Measure. Accepts the known D-spacing of any compound and calculates the position at which the arc should appear on an X-ray diffraction powder film of the compound.

"POWID" -- A searching system for identifacation of compounds using D-string and intensity pairs for X-ray Powder Difraction Measurements.

"RADIAL-1" -- Calculates and plots the radial distribution functions for the 1s,2s,sp,3s,3p, and 3d hydrogen-like atomic orbitals.

"RADIO" -- Calculates radio-decay curves.

"SOL-1" -- Calculates the solubility of the salt mnap as a function of ph.

"STOIC" -- Uses the basic equation xa+yb=zc+wd and calculates product and reactant amounts as well as determining the limiting reagent.

"VSEPR-1" -- Predicts the basic geometry of certain types of molecules and ions.

********* MATHEMATICS PROGRAMS *********

"CHAREQ" -- Finds the coefficients in the characteristic polynomial of a real matrix.

"ECHLN" -- Transforms a matrix to echelon form by elementary row operations.

"GRAM" -- Takes a basis of row vectors for a subspace of an n-tuple space, and computes an orthonormal basis of row vectors.

"JACOBI" -- Diagonalizes a real symmetric matrix.

"LINEAR" -- Does a linear fit of n data points.

"POLREG" -- Polynomial regression analysis routine using the method of least squares.

"REDECH" -- Transforms a matrix to reduced echelon form by elementary row operations.

"SIMPLEX" -- Solves a linear programming problem by the simplex method.

********** UTILITY PROGRAMS **********

"CARDFREE" -- Takes a card image Fortran program and reformats it into a free format Fortran suitable for DTSS Fortran.

"FREECARD" -- Takes a free format Fortran program in DTSS Fortran and formats it into a card image style program divided according to columns 1-5,6,7-72,73-80.

"FORMATS" -- Takes a card image Fortran program and moves all the FORMAT statements to just before the END statement sorted in order according to their statement numbers.

"DESEQ" -- Takes a card image Fortran program with sequence numbers in columns 73-80, and removes them along with all the trailing spaces.

"FTRAN" -- Takes a card image Fortran program and puts out a Basic program with comments on the lines that it was unable to translate.

"COMMENTS" -- A quick way of getting rid of comments from a Basic program that was created by FTRAN. Will remove comments only from lines specified by the user.

"LINELEN" -- Checks length of lines in users current file to avoid "record length errors" in writing tapes with fixed length records.

"SEQUENCE" -- Checks length of lines in the users current file and adds sequence numbers in columns 73-80.

APPENDIX L

CONDUIT Staff 1973-74

CONDUIT Staff

1 June 1973 - 31 May 1974

John M. Nevison	Project Manager
Christopher G. Hoogendyk	Chief Programmer and Acting Manager

Programmers

David Cooley - Program Librarian

Howard S. Becker
Andrew Behrens
David I. Chemerow
Steven Fellows
Bruce Flaherty
Mark Hagen
Carey Heckman
John C. Miller
Richard A. Pechter
David Roos
William R. Schillhammer

Slide Tape Producers

Josh Lee
Michelle Murray
Spencer Reiss
David G.S. Wood

Secretary

Linda M. Zoccoli
Linda Maynard

APPENDIX M

Working Paper #8

Programmer's Weapons

CONDUIT

Programmer's Weapons

John M. Nevison

A consortium of regional networks at
Oregon State University, North Carolina Educational Computing Service,
Dartmouth College, and the Universities of Iowa and Texas (Austin).

Programmer's Weapons

John M. Nevison

May 1974

Dartmouth-CONDUIT

Working Paper #8

```
*********************************************************************
* DRAFT COPY - Not to be reproduced without the written permission *
* of the author.                                                    *
*********************************************************************
```

Programmer's Weapons

For import or export you should have a copy of Kernigham and Plauger's The Elements of Programming Style. The Bookstore sells it for $2.95. Buy it. Read it.

Importing a Program

The goal of importing a program is to see it well used in a course. This means that after a program is on the system (See the Working Paper #7, for details.), it will be translated into BASIC.

If you are ready to begin, be sure you have in your hands

1. All the available information on the program
2. The name, address, and telephone number of the sender.

Working Paper #4 on Editorial Aids for Importing and Exporting of Fortran Programs on DTSS should help you out. Read it over, before you start.

Other working papers listed in the appendix may also be of help. Check them over, before you start.

After the program runs and produces the correct results with sample data,

1. Put a proper header on the program
   (Check Working Paper #1 for standard)
2. Complete the internal documentation of the program
   (as well as possible)

It may be that your program produces its correct results and is still a mystery to you. Don't worry about it. Imported programs are only guaranteed to work as advertised. CONDUIT can not doctor every under-explained program that is shipped in. That's the job of either the author or the professional (economist, chemist, sociologist) who uses it.

Give a copy of the finished program to the Chief Programmer and tell the librarian it is ready to be put into CONLIB.

Please keep notes on your work until you have put everything down on a Programmer's Report.

Exporting a Program Set

Start with the text that contains the programs. Think about how to write the programs according to Working Paper #1, Preparing a Program for Export.

1. -Make friends with the programs
2. -Structure them in good BASIC or Fortran (See Reference)
3. -Comment the programs
4. -Re-read the text
5. -Write one or several Q-files (See Working Paper #2, Describing a Program for Export)
6. -Check your programs and Q-files against the samples in the Working Papers for completeness.

Finish by turning them over to the Manager and Chief Programmer for proof reading.

Please, keep notes on your work until it is written down in a Programmer's Report.

APPENDIX

CONDUIT-Dartmouth Working Papers

1. Preparing a Program for Export, Christopher G. Hoogendyk, April 1974.
2. Describing a Program for Export, John M. Nevison, April 1974.
3. Agreeable Fortran, Christopher G. Hoogendyk, May 1974.
4. Editorial Aids for Importing and Exporting Fortran Programs on DTSS, Christopher G. Hoogendyk, October 1973.
5. The Export System, John M. Nevison, May 1974.
6. How Combib Works, John M. Nevison, May 1974.
7. Magnetic Tape Format for Exporting Programs, Christopher G. Hoogendyk, May 1974.
8. Programmer's Weapons, John M. Nevison, May 1974.
9. Isaacs, Gerald L. Interdialect Translatability of the Basic Programming Language, American College Testing Program, Bulletin No. 11, October 1972, available from CONDUIT Central, Iowa City, Iowa.
10. USA Standard Fortran (USAS X3.9-1966) United States of America Standards Institute, New York, 1966.

APPENDIX N

Knights in Rusting Armour

(Report without Appendices)

CONDUIT

Knights in Rusting Armour
(Report without Appendices)

A consortium of regional networks at
Oregon State University, North Carolina Educational Computing Service,
Dartmouth College, and the Universities of Iowa and Texas (Austin).

Knights in Rusting Armour
(Report without Appendices)

John M. Nevison

October 1973

27 October 1973

Knights in Rusting Armour; A Report on the CONDUIT Activities at Dartmouth, June 1972-May 1973 (Also known as Interim Report, Technical Report, or Regional Report)

John M. Nevison

Part 1--Introduction

The following report is dull. It will not detail any new uses of the computer, it will not relate any exciting incidents from the classroom during the past year, nor will it provide much insight into the attitude of any participant, student or teacher, after his run-in with the project called CONDUIT.

The reader interested in learning what went on in the classroom is urged to write to the participants. (See Appendix B) If the reader is hungry for more he should address his inquiries[*] to the Human Resources Research Organization (HumRRO) which supervised the evaluation of the project and is analysing the results of its own questionaires.

What this report will discuss in some detail is the first stumbling efforts of a small staff at one computer center as it attempted to translate imported programs into the local language and make these programs useful to teachers and students.

The bulk of the discussion to follow concerns problems that preoccupied the staff during the school year. It was a frustrating time. First, because it took the coordinator four months to realize that he was understaffed with only two part time student programmers. Second, because a lack of standards allowed incorrect and poorly documented programs to be shipped to the center. Third, because the discipline committees made hurried decisions and in several cases selected mediocre texts.

---

[*] Robert Seidel
HumRRO, Div.1, 300 N. Washington St., Alexandria, Va. 22314

It was also a rewarding time. Some programs arrived with exemplary documentation. After an increase to six student programmers at Dartmouth the pace of technical transport increased from shuffle to jog. Several of the teachers dramatically increased the use of computers in their courses.

In moving an idea from one classroom to another, moving computer programs played a small part. Before an idea moved there had to exist a donor and a recipient. During the past year the donors were the workshop teachers selected by the discipline committees of the project; the recipients, those who attended the workshop. These workshop participants were selected by the central office in consultation with the discipline committees.

The participants returned to try the new workshop ideas in their classes. Copies of computer programs that accompanied the ideas were sent from the computer center hosting the workshop to each of her four sisters. By the first of September Kiewit Computation Center at Dartmouth had received the following sets of computer programs.

--Physics programs from John Merrill's text, The Computer in Second Semester Physics. These programs were written originally on the Dartmouth System and were available from the author or the project office. Interested users were encouraged to copy these short programs from the book.
--Business and Economic programs from Oregon State University at Corvallis. These programs were well documented but not urgently needed. No one from the Dartmouth Region had been selected for the Business Workshop. The one teacher who attended the Economics Workshop did not plan to use the programs until March.
--Linear Algebra programs from the University of Iowa These well documented programs were translated into Dartmouth Fortran in one week and as quickly forgotten. The only teacher in the Dartmouth Region who had attended the Workshop returned to discover that his school had disconnected from the Dartmouth Computer.
--Chemistry programs from the University of Texas. The poorly documented, incorrect programs lay in their card decks like so many microbes ready to infest the system with mysterious errors. Two teachers wanted to use some of these programs within a month so translating them became the staff's first effort.
--Social Science programs did not exist. The Social Science Committee had not yet held its workshop. A local agreement with project IMPRESS (Interdisciplinary Machine Processing for Research and Education in the

TABLE 1

TECHNICAL COSTS (IN DOLLARS)
30 August 1972-30 May 1973

	COMPUTING COST	PROGRAMMER COSTS ($2.00/HR)	TOTAL TECHNICAL COST
Biology (not done)	-------	-------	-------
Business (incomplete when concluded)	342.18	169.50	511.68
Chemistry (incomplete when concluded)	1393.58	708.00	2101.58
Economics (complete)	292.99	184.50	477.49
Linear Algebra (complete)	60.06	57.00	117.06
Physics (complete)	00.00	00.00	00.00
Social Science (incomplete)	not available	320.75	320.75
Translation Programs (complete)	339.53	209.00	548.53

TABLE 2

PACKAGE COSTS (IN DOLLARS)
30 August 1972-30 May 1973

	TECHNICAL	LOCAL SUPPORT	CLASS USE	TOTAL
Biology	-------	-------	-------	-------
Business	511.68	-------	-------	511.68
Chemistry	2101.58	77.63 (student programmer)	792.64	2971.85
Economics	477.49	-------	493.85	971.34
Linear Algebra	117.06	822.00 (TTY rental)	not available	939.06
Physics	00.00	00.00	not available	00.00
Social Science	320.75	480.00 (student programmer)	1562.05	2362.80
Translation Programs	548.53	-------	-------	548.53
TOTAL	4077.09	1379.63	2848.54	8305.26
	49%	17%	34%	

Social Sciences) provided for cooperative efforts to translate appropriate texts.

The problem that would occupy the staff for the next nine months was set: help provide the teacher with computer programs he can use in his class.

On the Dartmouth Time-Sharing System (DTSS) almost everyone uses Basic. The programs that arrived during the summer were in Fortran. In order to allow students and teachers to alter easily a CONDUIT program, and to increase the likelihood of the wider adoption of a CONDUIT program, each one was translated from Fortran to Basic.

The cost of Table 1 is not the cost to move the program, but the cost to move the program well, that is, the cost to put a program on the computer and to translate it into Basic.

Even after this translation, it must be admitted, it was not accurate to describe a program as well moved. Many programs suffered from the inadequacies of their native batch processing environments. Translated into a good interactive environment they looked about as appropriate as a knight in full armour playing split end for the Dallas Cowboys.

Re-equiping a program for time-sharing was not the responsibility of the CONDUIT office. It was the responsibility of the individual teacher. The office provided student programmers to on-campus users and, at remote schools, money to teachers who hired their own student programmers.

This flexible teacher-student team provided the human analog to the flexible time-sharing system where a program could be tailored to suit either an individual class or an individual student.

The project office began with the urgently needed Chemistry programs and prepared each set of programs as they were needed. By the end of the year all useful programs and most of the selected programs had been translated.

Part2--The Programs by Disciplines

Translation Aids

A set of translation aids was urgently needed by the Project staff. This set of programs began in August of 1972 and underwent several changes during the project.

TABLE 3

CONDUIT CHEMISTRY PROGRAMS 1972-1973

BOYLES-GILLETTE PROGRAM SELECTION (in order)	JOHNSON'S TEXT'S PROGRAMS	JOHNSON, DENK, & COLLIN'S PROGRAMS
1 STOIC	1 ABC	* 1 POWAC
+ 2 HATOM	+ 2 BOX	* 2 POWID
+ 3 RADIAL	+ 3 CONTOUR	3 STOIC
+ 4 CONTOUR	+ 4 EDTA	+ 4 IONCH
+ 5 VSEPR	5 ENTROPY	+ 5 HATOM
+ 6 IONCH	+ 6 EQUIL	+ 6 MASSPEC
+ 7 EQUIL	+ 7 HA	7 FRSTLW
+ 8 EDTA	8 HMO	* 8 ONELEC
+ 9 HA	9 IR	9 DERTI
+10 SOL	10 JACOBI	10 IDFOU
+11 NERNST	11 LINEAR	+11 RADIAL
*12 KNEXP	*12 MATINV	+12 VSEPR
13 A=B=C	+13 NERNST	13 A=B=C
+14 MASSPEC	*14 NEWTON	*14 KNEXP
15 IDGAME	15 NMR	15 RADIO
16 ABC	*16 NONLIN	16 IDGAME
17 NMR	*17 PLOT	*17 SEIDEL
18 DERTI	18 POLREG	*18 SIMP
*19 NEWTON	*19 RUNGE	*19 SIMQ
20 RADIO	*20 SECANT	*20 CHEM200
21 FRSTLW	+21 SOL	*21 CHEM32
+22 BOX		
23 HMO		

+ Programs used in class

* Programs not successfully translated

Underlined Programs selected by CONDUIT Chemistry Discipline Committee

The initial efforts during the fall were not very successful. The programmer who wrote TRAN, a large program that translated much of a Fortran program into Basic, was not readily available to his fellow programmer who discovered many of TRAN's bugs while working on the Chemistry programs.

By the end of February a new chief programmer and a new program, FTRAN, were both working splendidly. Five new programmers could telephone the chief programmer at home and he would immediately fix any bugs in FTRAN from the terminal at his home. In addition to FTRAN, the chief programmer wrote a collection of editorial aids for the translation and the formatting of programs. A draft of a Computer Center Publication describing those aids is enclosed as Appendix C.

The total cost of these programs was $548.53. This cost was included in the total technical costs ot the year in Table 2 because without the translating programs the other work would have been impossible.

Chemistry

> "IONCH-Clarity of the program and its usage was seen as lacking" (From the Chemistry Workshop Report)

Table 3 illustrates the year's success and failures in Chemistry. The Project office initially concentrated on the list of 23 programs that Gillette and Boyles requested. After completing these programs, the office concentrated on completing the translation of all programs in K. Jeffrey Johnson's Numerical Methods in Chemistry. (Second Edition). When the book was complete the leftovers were worked on. Some were completed and some were not.

Among the entries in Table 3 are some of the best and the worst programs in this year's work. The worst sins were programs like CONTOUR, a program that looked good but would not run correctly. It did not run correctly at the workshop and did not run correctly before it was sent from the University of Texas to Dartmouth. Worse than the fact that it did not run was the omission of any warning that it did not run. The outstanding student programmer on the Dartmouth staff wasted forty hours trying to debug a program with no bugs. The program did what it was supposed to do: it correctly evaluated five (5) incorrect equations. That a committee could select such a program for transport is a disgrace.

The next sin of program transport is failure to send sample data and a listing of how the program should run using this data. KNEXP, POWAC, and POWID after repeated requests beginning in April of 1973 (Seven months *after* the workshop) and extending through the end of May *still* could not be verified at Dartmouth. Again, the Chemistry Discipline Committee's selection of unverified programs was responsible for wasted efforts, unfulfilled expectations, and damaged reputations.

The third sin of program transport is a program with inadequate directions on use. The efforts of this year have illustrated an obvious truth: computer center personnel are in no position to describe how to use programs in courses. The natural consequence of this truth, *that programs need adequate directions on use before they are selected*, escaped the dull eyes of the Discipline Committee. *At no time during the year did they see fit to correct their initial mistakes*, and discard bad or poorly documented work. Nor did the Central Office see fit to encourge the committee to monitor the use of its programs. (Monitor, meant making phone calls twice during the year to check with workshop participants and coordinators on the worth of the selected programs).

A fourth sin of transport is a program without a flowchart, or a program without good internal documentation. Some selections were so poorly documented that students could not improve the CONDUIT selection to meet the simple standards of the DTSS program library.

The Chemistry Committee atoned for many of its sins by selecting one of the outstanding collection of programs involved in the first year's activities. K. Jeffrey Johnson's *Numerical Methods in Chemistry* was a delight to work with. There was proper description of the subject and full documentation of all programs in the book.* This documentation included flow charts, listings, sample data and runs, and references on the procedures employed.

As the Project staff worked its way down the list of desired items it encountered another simple problem, not all the programs were sent by Texas. So on November 21, 1972, NEWTON, ONELEC, ENTROPY, CHEM200, CONTOUR, IONCH, POWAC, POWID, KNEXP, IDGAME, POLREG, and SEIDEL were requested from Texas. They were promptly shipped Nov. 27.

---

*The *Second Edition* which arrived at the end of February contained a correct version of CONTOUR.

For many of the above sins of transport, the project paid with wasted time and frustrated effort. Teachers and students also paid. Many programs were late for class. Some cut the entire course.

Another problem aluded to earlier was making the shift from batch to interactive programs. *The only person qualified to make this translation was the teaching professor.* James Boyles at Bates College paid his student programmer $77.63. He himself contributed *many* additional hours of effort to improve programs for use in his class (See the programs labeled "-1" in LIBCAT Appendix D.) Robert Gillette at the Naval Academy was not allowed to hire cadets and so he also spent *many* hours of time and effort tayloring programs to his class's needs. These hours are *not* in Table 2.

Table 3 shows that fifteen (15) programs were not successfully translated during the year of classroom experimentation. Nine (9) of these programs were numerical routines with analogous programs already existing in the DTSS program library. Six (6) either did not work or lacked sample data and runs to ensure their accuracy. Twenty-seven (27) programs were successfully translated. Twelve (12) were used by Boyles and Gillette.

Table 2 lists the total computer bill for Gillette's and Boyle's courses at $792.64. This figure is *not* an accurate reflection of the total use of the Chemistry programs during the past year. Several teachers at Skidmore College modified the Chemistry programs and used them in their courses. Even with the excessively large cost of translation it would be incorrect to assume that it overshadowed the class use of the programs. The cost of translation was a one-time cost and those programs with adequate documentation, that is, those in Jeff Johnson's book, will be used in succeeding years.

Casting a final glance at Table 3, one must note in conclusion that there remain a number of programs, translated but unused, batch programs in an interactive world, that stand like so many knights rusting in their armour watching a game they only dimly comprehend.

Economics

One of the first tasks to confront the enlarged team of programmers that began work in late February was the set of Economics programs.

The programs that were useful to George Michaelides, BULLPUP and MCWHARTON, were not available for his class. Some of the other programs, notably, FACTOR-FACTOR II, and SIMPLE FIRM, were impossible to run correctly and were happily discarded when instructed to by CONDUIT Central. The Economics committee subsequently withdrew its endorsement of several sloppy programs. Most of the time spent on the economics package was time wasted on poorly written programs. This situation could have been avoided if the discipline committee had insisted on seeing running versions of the programs (verified programs) before approving them for CONDUIT distribution.

The programs that were used by Michaelides have been neglected by the other users of DTSS. This neglect is largely the result of the total absense of an adequate description of how to use these programs in an Economics course.

In late February with the publication of a sheet entitled Workshop Materials the Dartmouth Office was informed, much to its surprise, that an IMPRESS-like workbook entitled MACROECONOMICS by Foster Mattson was part of the set of Economics programs. (See Appendix D.) To date no professor in the Dartmouth Region has expressed any interest in supervising the translation of this workbook for the users of DTSS.

The small flame lit by the Project during the first year may well flicker and die during the second. The only hope is that an adequate and appealing new text will appear and that this text will catch the interest or a local economist.

Business

The Business programs were play for the Project staff. One programmer having completed his other work, spent a few hours bringing up all the programs in the book, Computer Models in Operations Management.

After these programs were completed a professor at the Amos Tuck School of Business Administration grew interested in supervising the translation of additional programs from the second text in the business set, Computer Augmented Cases in Operations and Logistics Management.

Like the Chemistry programs, a great deal of modification was necessary before the Business programs were acceptable for use in classes. In some cases, considerably more sophisticated programs already existed in the DTSS program library.

The programs stand an excellent chance of being used during the coming year because they have a local sponsor and because they are all associated with a text that provides a reasonable discription of how to use the programs.

Linear Algebra

The programs in Donald McLaughlin's A Computer Oriented Course in Linear Algebra were easy to handle. In August of 1972 they were translated into Dartmouth Fortran in five hours. In March a programmer spent 23.5 more hours to translate them into Basic and provide card decks to Barrien Moore.

Moore visited Hanover to speak to a conference about how his students had reacted to McLaughlin's approach. His enthusiasm was catching and others, including some of the Dartmouth Mathematics Department, expressed interest in using some ideas in their own courses.

Moore employed his own student programmer but received money to rent a teletype and telephone for his students to use during the course.

The programs running now stand an excellent chance of being used this year if others adopt the McLaughlin text.

Physics

No physics programs were translated during the year. All existed at the start and ran in Basic. The Dartmouth Region participants were at the University of New Hampshire and used the UNH computer exclusively. They neither asked for, nor received, additional help from the Project office.

Biology

Two copies of David Wilson's book arrived in May of 1973. No programs were translated. Errors in the programs in the text were discovered by both a programmer and the professor who reviewed the book.

Because the description of how to use the programs is poorly organized, the text will probably not be used in the year 1973-1974, in the Dartmouth Region.

Social Science

The Social Science activities of project CONDUIT were supervised by the IMPRESS office under the direction of Edmund D. Meyers, director of Project IMPRESS.

The original aim of this project was to translate six of the nine recommended workbooks. Two of these workbooks were available at the time of the October 1972 Social Science Workshop. No more were completed before the end of the year. (See Appendix D.)

It is interesting to note that in the Coordinator's eyes the Project failed because it failed to produce two additional workbooks desired by Rosenberg and Hartshorn. In the Discipline Committee's eyes the Dartmouth Region failed because all nine workbooks were not translated. At no time did anyone ever define the translation of all nine workbooks as the Region's responsibility, or the Project's goal.

Given the workbook's elementary approach, it would have been a waste of time and effort to support any more than the minimum deemed desirable by DTSS faculty. (G.R. Boyton worked all the excerises in one workbook in an hour on the IMPRESS system.)

During the last half of the year the Project paid a student programmer to help prepare two additional workbooks. (See Table 1) In addition, the Project supported David Rosenberg's student assistants at Middlebury. (See Table 2)

The $1562.28 of computer time under Class Use in Table 2 was not all the use of the CONDUIT databases. Other users of IMPRESS combined with the CONDUIT students to tally 1070/19002, or 5.6%, of the total IMPRESS runs recorded for the year June 25, 1972 to June 23, 1973.

Given the enormous range of the databases available on IMPRESS and given the wide variety of sophisticated techniques available for analysis, the solid use of the three (3) CONDUIT databases should underscore the power of an adequate written description of how to use a program in class, no matter how elementary.

This respectable usage should encourage the translation of additional elementary texts. David Rosenberg's informative report on his experience was completed on July 19, 1973. (See Appendix A)

Since June two more of the original workbooks have been completed. In addition, David Rosenberg, working over the summer with student help, is preparing additional workbooks for his own courses during the coming year. Elinor Hartshorn has also developed plans to use the new workbooks in her courses during the coming year.

Part 3--Local Workshops

Three workshops were held on December 9, March 16, and April 15 of the year 1972-1973. Full reports have been filed with CONDUIT Central (See Appendix A.) These workshops encouraged wider use of the summer's imported ideas. Speakers included four of the eight National Workshop Participants and many other active teachers in the Dartmouth Region.

Part 4-- National Physics Workshop

The National Physics Workshop ran from June 19 to June 30, 1972 at Dartmouth. It was taught by John Merrill using his own text, The Computer in Second Semester Physics. A full report has been filed with CONDUIT Central. (See Appendix A)

Part 5--Independent Swaps and Conclusions

Independent Swaps

Beginning in October of 1972 the Coordinators, recognizing that programs of quality existed that were not officially included in Project CONDUIT, began an informal exchange of programs. The details are elaborated in the Dartmouth Coordinator's memo of October 24, 1972. (Enclosed as Appendix E.)

Of the eight sets listed, two were from the Dartmouth Region; Ecological Modeling on the Dartmouth Time-Sharing System by Glanz and Reiners and J. Peter Williamson's Investments: New Analytic Techniques (Praeger) with its partner Manuals for Computer Programs in Finance and Investments. Both were shipped promptly. In early

November Praeger was requested to send desk copies of Williamson's text to all coordinators.

The Reiner's programs were adopted at Texas and Oregon. Subsequent errors discovered in the biology text caused Reiners to immediately halt distribution. Chris Hoogendyk working with Reiners corrected the mistakes and, after an April to October delay, the text was again approved for distribution.

J. Peter Williamson's text was not, to the knowledge of the Dartmouth Coordinator, adopted by any other Project office.

Of the remaining six program sets listed for swapping, Dartmouth never requested or received two; Exper-Sim and Naylor's Simulation of State Government Dartmouth reviewed but did not choose to accept OSU's graphing system because of the machine language subroutines and Snyder's Statistical Method's in Biology because no one here was teaching that kind of course at the time. Both will be reconsidered this year.

At the joint request of David Baldwin in the Government Department at Dartmouth and David Rosenberg in the Department of Political Science at Middlebury both PRINCE from Sycause University and INS from North Carolina have been translated and now work on the Dartmouth Time-Sharing System. PRINCE and INS work began in May of 1973 and the work on these programs falls outside the scope of the present report.

It must be noted in passing that in addition to the independent swaps, publicity work, and regular classroom support, the Project Office engaged in a minor, and completely unsuccessful, trial distribution designed by HumRRO. A full report of that trial was made in August 1973 (See Appendix A).

Conclusions

The experience of the first year has, first, defined a necessary minimum for program transport. Second, it has suggested what an outstanding set of programs should look like. Third, it has underscored the gross differences that exist between the different computing environments in the project. Fourth, it has sketched a preliminary idea for necessary standards.

A working program needs the blessing of a wise selection by the discipline committee and a perfunctory performance by the Project programming staffs. When the discipline committee approved bad programs, the translation failed. When the discipline approved programs that did not have sample data and sample runs, the translation failed. When the programming staff at Dartmouth was undermanned the translation was intolerably delayed.

If the discipline committee selects wisely and the Coordinator's work is adequate, a program may still not be translated. Several programs in one Business text were ignored by the Dartmouth Office. The above conditions are not sufficient. Three good working definitions of what a sufficient set of programs should be are:

K. Jeffrey Johnson's, Numerical Methods in Chemistry
Donald McLaughlin's, A Computer Oriented Course in Linear Algebra
John Merrill's, The Computer in Second Semester Physics

All of these examples are texts. They provide ample description of how to use the computer programs in class by providing a detailed discussion of the subject. In addition to a description of how to use the programs in class, each book contains programs that have adequate internal documentation and, where necessary, program flow charts. A third important characteristic of all three examples is that each includes sample runs of the programs and sample data where appropriate. Fourth and most important, because these sets were good texts they were adopted by an enthusiastic, qualified teacher.

This qualified, enthusiastic teacher is both necessary and sufficient to carry translated programs into his class. In an interactive time-sharing environment a qualified teacher is one literate in computing. He knows how to program, not necessarily to do much of it, but to smoothly supervise his student programmers.

Even texts sufficiently prepared and adequately translated did not always get adopted by a qualified teacher in the Dartmouth Region. The flexible, easy to use, time-sharing system often mocked the batch programs original hope of saving someone time and effort. Many programs translated by the project could have been rewritten from scratch and been twice as useful at half the cost.

In other words the project could have been four times as effective by simply mailing texts to teachers in the Dartmouth Region and giving each teacher an ample budget to hire students to do his programming. Four is a conservative multiplier and the real increase in benefits might be eight to ten times the current scheme. This conclusion underscores the enourmous difference between an adequate educational computing environment and the systems currently operating at some of the other Project centers.

Because of the variety of educational environments, equipment, languages, and character codes, standards are needed within the Project. Experience of the past year has informed several publications soon to be released by the Project.*

These documents should specify standard character codes, standard magnetic tape procedures, common language conventions, common staff programmer procedures, and standard review guides.

The past year's activity has demonstrated how not to transport a program. It has demonstrated how a program might be transported successfully. It has sharpened the understanding of how a computer center publicizes good ideas. It has underscored the differences among computing environments. And finally, it has suggested several necessary steps to harmonize the future exchange of programs.

*For information contact:
James Johnson, Director
Project CONDUIT
Regional Computer Center
University of Iowa, Iowa City, Iowa 52240

APPENDICES

(Not included)

A. Reports from the Dartmouth Region June 1972-October 1973

B. Dartmouth Region Participants in Classroom Test of Workshop Ideas

C. Draft of a Local Publication on Translation Aids

D. Local Information on Project CONDUIT

E. Program Swaps Memo

F. Project Staff 1972-1973

APPENDIX O

Dartmouth-CONDUIT

June 1, 1974 - December 31, 1974